Lecture Notes in Computer Science 5229

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Rafail Ostrovsky Roberto De Prisco
Ivan Visconti (Eds.)

Security and Cryptography for Networks

6th International Conference, SCN 2008
Amalfi, Italy, September 10-12, 2008
Proceedings

Springer

Volume Editors

Rafail Ostrovsky
University of California, Los Angeles
Department of Computer Science
Box 951596, 3732D BH, Los Angeles, CA, 90095-1596, USA
E-mail: rafail@cs.ucla.edu

Roberto De Prisco
Università di Salerno
Dipartimento di Informatica ed Applicazioni
via Ponte don Melillo, 84084 Fisciano (SA), Italy
E-mail: robdep@dia.unisa.it

Ivan Visconti
Università di Salerno
Dipartimento di Informatica ed Applicazioni
via Ponte don Melillo, 84084 Fisciano (SA), Italy
E-mail: visconti@dia.unisa.it

Library of Congress Control Number: 2008933864

CR Subject Classification (1998): E.3, C.2, D.4.6, K.4.1, K.4.4, K.6.5, F.2

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-540-85854-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-85854-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12512393 06/3180 5 4 3 2 1 0

Preface

The 6th Conference on Security and Cryptography for Networks (SCN 2008) was held in Amalfi, Italy, on September 10–12, 2008. The first four editions of the conference where held in Amalfi, while, two years ago, the fifth edition was held in the nearby Maiori. This year we moved back to the traditional location.

Security and privacy are increasing concerns in computer networks such as the Internet. The availability of fast, reliable, and cheap electronic communication offers the opportunity to perform, electronically and in a distributed way, a wide range of transactions of a most diverse nature. The conference brought together researchers in the fields of cryptography and security in communication networks with the goal of fostering cooperation and exchange of ideas. The main topics of the conference this year included anonymity, implementations, authentication, symmetric-key cryptography, complexity-based cryptography, privacy, cryptanalysis, cryptographic protocols, digital signatures, public-key cryptography, hash functions, identification.

The international Program Committee consisted of 24 members who are top experts in the conference fields. The PC received 71 submissions and selected 26 papers for presentation at the conference. These proceedings include the 26 accepted papers and the abstract of the invited talk by Shai Halevi.

The PC selected papers on the basis of originality, quality and relevance to the conference scope. Due to the high number of submissions, paper selection was a difficult task and many good papers had to be rejected. Each paper was refereed by three or four reviewers. We thank the members of the PC for the effort invested in the selection process. We also gratefully acknowledge the help of the external reviewers who evaluated submissions in their area of expertise. The names of these reviewers are listed on page VIII, and we apologize for any inadvertent omissions or mistakes.

Finally, we would like to thank the authors of all submitted papers and the conference participants, who ultimately made this conference possible.

September 2008

R. Ostrovsky
R. De Prisco
I. Visconti

SCN 2008

September 10–12, 2008, Amalfi, Italy

Program Chair

Rafail Ostrovsky	University of California, Los Angeles, USA

General Chairs

Roberto De Prisco	Università di Salerno, Italy
Ivan Visconti	Università di Salerno, Italy

Program Committee

Carlo Blundo	Università di Salerno, Italy
Xavier Boyen	Voltage Inc, USA
Ran Canetti	IBM, USA
Dario Catalano	Università di Catania, Italy
Ronald Cramer	CWI & Leiden University, The Netherlands
Serge Fehr	CWI, The Netherlands
Juan Garay	Bell Labs - Alcatel-Lucent, USA
Rosario Gennaro	IBM, USA
Jens Groth	University College London, UK
Yuval Ishai	Technion and UCLA, Israel and USA
Jonathan Katz	University of Maryland, USA
Eyal Kushilevitz	Technion, Israel
Ueli Maurer	ETH Zurich, Switzerland
Daniele Micciancio	UCSD, USA
Phong Nguyen	ENS, France
Tatsuaki Okamoto	NTT Laboratories, Japan
Rafail Ostrovsky (chair)	UCLA, USA
Giuseppe Persiano	Università di Salerno, Italy
Benny Pinkas	University of Haifa, Israel
Tal Rabin	IBM, USA
Leonid Reyzin	Boston University, USA
Adi Rosen	CNRS and University of Paris 11, France
Adam Smith	Pennsylvania State University, USA
Ivan Visconti	Università di Salerno, Italy

Referees

Divesh Aggarwal
Zuzana Beerliova
Charles Bouillaguet
Suresh Chari
Debbie Cook
Cécile Delerablée
Mario Di Raimondo
Orr Dunkelman
Dario Fiore
Sebastian Gajek
David Galindo
Peter Gaži
Craig Gentry
Sharon Goldberg
Amir Herzberg
Alejandro Hevia
Dennis Hofheinz
Susan Hohenberger
Emeline Hufschmitt
Charanjit Jutla
Bhavana Kanukurthi
Aggelos Kiayias
Eike Kiltz
Vladimir Kolesnikov
Gaëtan Leurent
Anna Lysyanskaya
Vadim Lyubashevsky
Alexander May
Lorenz Minder
David Molnar
Christopher Portmann
Emmanuel Prouff
Dominik Raub
Mike Rosulek
Amit Sahai
Christian Schaffner
Nigel Smart
Stefano Tessaro
Carmine Ventre
Enav Weinreb
Daniel Wichs
Vassilis Zikas
Cliff Changchun Zou

Table of Contents

Session 5: Primitives

Session 6: Signatures

Session 7: Hardware and Cryptanalysis

Session 8: Protocols II

Session 9: Encryption III

Session 10: Key Exchange

Storage Encryption: A Cryptographer's View

Shai Halevi

IBM Research, Hawthorne, NY, USA
shaih@alum.mit.edu

Abstract. Encryption is the bread-and-butter of cryptography, with well-established notions of security and a large variety of schemes to meet these notions. So what is left for researchers in cryptography to look at when it comes to encrypting storage? In this talk I will cover cryptography issues that arise when introducing encryption to real-world storage systems, with some examples drawn from the work of the IEEE 1619 standard committee that deals with standardizing aspects of storage encryption. The issues that I plan to touch upon include:

Encryption Schemes and Modes-of-Operation: The use of "authenticated" vs. "transparent" encryption, "wide block" vs. "narrow block" transparent encryption modes, and other considerations.

Issues with Key-Management and IV-Management: How to avoid nonce collision when your nonces are only 96-bit long, why you may want to use deterministic encryption for key-wrapping, what is the difference between key-wrapping and KEM/DEM, and related questions.

Self-Encryption of Keys: Can an encryption scheme remain secure when used to encrypt its own secret key? It turns out that this requirement sometimes comes up when encrypting storage. I will talk about several aspects of this problem, including the not-so-bad, the bad, and the ugly.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, p. 1, 2008.

Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries*

Yehuda Lindell[1], Benny Pinkas[2], and Nigel P. Smart[3]

[1] Dept. Of Computer Science,
Bar Ilan University,
Ramat Gan, Israel
lindell@cs.biu.ac.il
[2] Dept. of Computer Science,
University of Haifa
Haifa 31905, Israel
benny@pinkas.net
[3] Dept. Computer Science,
University of Bristol,
Woodland Road, Bristol, BS8 1UB, United Kingdom
nigel@cs.bris.ac.uk

Abstract. We present an implementation of the protocol of Lindell and Pinkas for secure two-party computation which is secure against malicious adversaries [13]. This is the first running system which provides security against malicious adversaries according to rigorous security definition and without using the random oracle model. We ran experiments showing that the protocol is practical. In addition we show that there is little benefit in replacing subcomponents secure in the standard model with those which are only secure in the random oracle model. Throughout we pay particular attention to using the most efficient subcomponents in the protocol, and we select parameters for the encryption schemes, commitments and oblivious transfers which are consistent with a security level equivalent to AES-128.

1 Introduction

Secure multi-party computation is a process which allows multiple participants to implement a joint computation that, in real life, may only be implemented using a trusted party. The participants, each with its own private input, communicate without the help of any trusted party, and can compute any function

* The first author was supported by The Israel Science Foundation (grant No. 781/07) and by an Infrastructures grant from the Israeli Ministry of Science. The other authors were supported by the European Union under the FP7-STREP project CACE. The second author was also supported by The Israel Science Foundation (grant No. 860/06).

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 2–20, 2008.

without revealing information about the inputs (except for the value of the function). A classic example of such a computation is the Millionaires' problem, in which two millionaires want to know who is richer, without revealing their actual worth.

Multi-party computation has been considered by the theoretical cryptography community for a long time, starting with the pioneering work of Yao [24] in 1986. Yao's garbled circuit construction is relatively simple, and runs in a constant number of rounds. Yao's construction still remains the most attractive choice for generic secure two-party computation.

In recent years attention has focused on whether the theoretical work has any practical significance. In the two-party case the main contribution has been the FairPlay compiler [15], which is a generic tool translating functions written in a special high-level language to Java programs which execute a secure protocol implementing them. There are two major drawbacks with the current FairPlay implementation. Firstly it only provides weak security against malicious adversaries (where reducing the cheating probability to $1/k$ requires increasing the overhead by a factor of k), and has no proof of security (in particular, it is clear that it cannot be proven secure under simulation-based definitions). As such, its usage can only be fully justified for providing security against honest but curious (aka semi-honest) adversaries.[1] Secondly it does not make use of the latest and most efficient constructions of its various component parts.

In recent years the theoretical community has considered a number of ways of providing a variant of Yao's protocol which is secure against malicious adversaries. In the current paper we examine one of the more recent and efficient protocols for providing security for Yao's protocol against malicious adversaries, namely the protocol of Lindell and Pinkas [13] which is proved to be secure according to a standard simulation based definition, and as such can be securely used as a primitive in more complex protocols (see [8, Chapter 7], which in turn follows [6]).

Our work presents the following contributions:

- We provide an efficient implementation of the protocol of [13], which is secure against malicious adversaries. This is, to our best knowledge, the first implementation of a generic two-party protocol that is secure against malicious adversaries according to a standard simulation based definition. The implementation demonstrates the feasibility of the use of such protocols.
- We derive a number of optimizations and extensions to the protocol and to the different primitives that it uses. Unlike prior implementations we pay particular attention to using the most efficient constructions for the various components. For example we use elliptic curve based oblivious transfer protocols instead of finite field discrete logarithm based protocols.

[1] The cryptographic community denotes adversaries which can operate arbitrarily as "malicious". Semi-honest (or honest but curious) adversaries are supposed to follow the protocol that normal users are running, but they might try to gain information from the messages they receive in the protocol. It is, of course, easier to provide security against semi-honest adversaries.

– We also examine the difference between using protocols which are secure in the random oracle model (ROM) and protocols in the standard model.[2] Of particular interest is that our results show that there appears to be very little benefit in using schemes which are secure in the ROM as opposed to the standard model.[3]

1.1 Related Work

Research on security against malicious adversaries for computationally secure protocols started with the seminal GMW compiler [9]. As we have mentioned, we base our work on the protocol of [13], and we refer the reader to that work for a discussion of other approaches for providing security against malicious adversaries (e.g., [14,11,23]). We note that a simulation based proof of security (as in [13]) is essential in order to enable the use of a protocol as a building block in more complex protocols, while proving the security of the latter using general composition theorems like those of [6,8]. This is a major motivation for the work we present in this paper, which enables efficient construction of secure function evaluation primitives that can be used by other protocols. (For example, the secure protocol of [2] for finding the k^{th} ranked element is based on invoking several secure computations of comparisons, and provides simulation based security against malicious adversaries if the invoked computations have a simulation based proof. Our work enables to efficiently implement that protocol.)

The first generic system implementing secure two-party computation was FairPlay [15], which provided security against semi-honest adversaries and limited security against malicious adversaries (see discussion above). FairPlayMP is a generic system for secure *multi-party* computation, which only provides security against semi-honest adversaries [3]. Another system in the multi-party scenario is SIMAP, developing a secure evaluation of an auction using general techniques for secure computation [5,4]. It, too, supports only security against semi-honest adversaries.

1.2 Paper Structure

Section 2 introduces Yao's protocol for secure two-party computation, while Section 3 presents the protocol of [13] which is secure against malicious adversaries. Section 4 presents the different efficient sub-protocols that we used. Finally, Section 5 presents the results of our experiments.

[2] A random oracle is a function which is modeled as providing truly random answers. This abstraction is very useful for proving the security of cryptographic primitives. However, given any specific implementation of a function (known to the users who use it), this assumption no longer holds. Therefore it is preferable to prove security in the standard model, namely without using any random oracle.

[3] This is surprising since for more traditional cryptographic constructions, such as encryption schemes or signature schemes, the random oracle constructions are almost always twice as efficient in practice compared to the most efficient standard model schemes known. Part of the reason for the extreme efficiency of our standard model constructions is our use of a highly efficient oblivious transfer protocol which reduces the amortized number of zero-knowledge proofs which are required to be performed.

2 Yao's Garbled Circuit

Two-party secure function evaluation makes use of the famous garbled circuit construction of Yao [24]. In this section we briefly overview the idea. Note, however, that the following basic protocol is not secure against malicious adversaries, which is why the advanced protocol in the next section is to be preferred. The basic idea is to encode the function to be computed via a Binary circuit and then to securely evaluate the circuit on the players' inputs.

We consider two parties, denoted as P_1 and P_2, who wish to compute a function securely. Suppose we have a simple Binary circuit consisting of a single gate, the extension to many gates given what follows is immediate. The gate has two input wires, denoted w_1 and w_2, and an output wire w_3. Assume that P_1 knows the input to wire w_1, which is denoted b_1, and that P_2 knows the input to wire w_2, which is denoted b_2. We assume that each gate has a unique identifier Gid (this is to enable circuit fan out of greater than one, i.e. to enable for the output wire of a gate to be used in more than one other gate). We want P_2 to determine the value of the gate on the two inputs without P_1 learning anything, and without P_2 determining the input of P_1 (bar what it can determine from the output of the gate and its own input). We suppose that the output of the gate is given by the function $G(b_1, b_2) \in \{0, 1\}$.

Yao's construction works as follows. P_1 encodes, or garbles, each wire w_i by selecting two different cryptographic keys k_i^0 and k_i^1 of length t, where t is a computational security parameter which suffices for the length of a symmetric encryption scheme. In addition to each wire it associates a random permutation π_i of $\{0, 1\}$. The garbled value of the wire w_i is then represented by the pair $(k_i^{b_i}, c_i)$, where $c_i = \pi_i(b_i)$.

An encryption function $E^s_{k_1,k_2}(m)$ is selected which has as input two keys of length t, a message m, and some additional information s. The additional information s must be unique per invocation of the encryption function (i.e., used only once for any choice of keys). The precise encryption functions used are described in Section 4.1. The gate itself is then replaced by a four entry table indexed by the values of c_1 and c_2, and given by

$$c_1, c_2 : E^{\mathrm{Gid}\|c_1\|c_2}_{k_1^{b_1}, k_2^{b_2}} \left(k_3^{G(b_1,b_2)} \| c_3 \right),$$

where $b_1 = \pi_1^{-1}(c_1)$, $b_2 = \pi_2^{-1}(c_2)$, and $c_3 = \pi_3(G(b_1, b_2))$. Note that each entry in the table corresponds to a combination of the values of the input wires, and contains the encryption of the garbled value corresponding to these values of the input wires, and the corresponding c value. The resulting look up table (or set of look up tables in general) is called the Garbled Circuit.

P_1 then sends to P_2 the garbled circuit, its input value $k_1^{b_1}$, the value $c_1 = \pi_1(b_1)$, and the mapping from the set $\{k_3^0, k_3^1\}$ to $\{0, 1\}$ (i.e. the permutation π_3). P_1 and P_2 engage in an oblivious transfer (OT) protocol so that P_2 learns the value of $k_2^{b_2}, c_2$ where $c_2 = \pi_2(b_2)$. P_2 can then decrypt the entry in the look up table indexed by (c_1, c_2) using $k_1^{b_1}$ and $k_2^{b_2}$; this will reveal the value of

$k_3^{G(b_1,b_2)} \| c_3$ and P_2 can determine the value of $G(b_1, b_2)$ by using the mapping π_3^{-1} from the set c_3 to $\{0, 1\}$.

In the general case the circuit consists of multiple gates. P_1 chooses random garbled values for all wires and uses them for constructing tables for all gates. It sends these tables (i.e., the garbled circuit) to P_2, and in addition provides P_2 with the garbled values and the c values of P_1's inputs, and with the permutations π used to encode the *output* wires of the circuit. P_2 uses invocations of oblivious transfer to learn the garbled values and c values of its own inputs to the circuits. Given these values P_2 can evaluate the gates in the first level of the circuit, and compute the garbled values and the c values of the values of their output wires. It can then continue with this process and compute the garbled values of all wires in the circuit. Finally, it uses the π permutations of the output wires of the circuit to compute the real output values of the circuit.

Traditionally, for example in hardware design, one uses circuits which are constructed of simple gates which take at most two inputs and produce as most one output. In a Yao circuit a gate which takes n inputs and produces m outputs is encoded as a look up table which has 2^n rows, each consisting of a string of $O(m \cdot t)$ bits (where t is the security parameter which denotes the length of a key). Hence, it is often more efficient to use non-standard gates in a Yao circuit construction. For example a traditional circuit component consisting of k 2-to-1 gates, with n input and m output wires can be more efficiently encoded as a single n-to-m gate if $4k > 2^n$. In what follows we therefore assume the more suitable n-to-m gate construction. The extension of the above gate description to this more general case is immediate.

3 The Lindell-Pinkas Protocol

The protocol was presented in [13] and was proved there to be secure according to the real/ideal-model simulation paradigm [6,8]. The proof is in the standard model, with no random oracle model or common random string assumptions. We describe below the protocol in some detail, for full details see [13]. We remark that this description is not essential in order to understand the results of our paper. The important things to note are the basic structure of the protocol, as described in the next paragraph, and the fact that the protocol is based on the use of different types of commitments (statistically binding, statistically hiding, and computational), and of an oblivious transfer protocol. We describe the implementation of these primitives in Section 4.

The basic structure of the protocol: The protocol proceeds in the following steps. It has statistical security parameters s_1 and s_2. We replace P_2's input wires with a new set of $O(s_2)$ input wires, and change the original circuit by adding to it a new part which translates the values of the new input wires to those of the original wires. Then P_1 generates s_1 copies of Yao circuits and passes them to P_2, along with $O(s_1^2)$ commitments to the inputs. The input decommitments for P_1's inputs are transferred to P_2 via a batched oblivious transfer. Finally, after executing a number of cut-and-choose checks on the transferred circuits and

commitments, P_2 evaluates half of the circuits and determines the output value as the majority value of the outputs of these circuits. One of the contributions of this paper is to examine each of the above operations in turn and optimize the parameters and components used in the Lindell-Pinkas description.

3.1 The Protocol in Detail

As explained in [13] it suffices to present a protocol for the case where the output is learnt by P_2 and P_1 learns nothing. We consider the computation of $f(x, y)$ where P_1's input is $x \in \{0,1\}^n$ and P_2's input is $y \in \{0,1\}^n$.

The protocol is parameterized by two statistical security parameters s_1 and s_2. (In [13] these are a single statistical security parameter but we shall see later that in order to optimize performance these parameters really need to be treated separately.) The protocol takes as input a circuit description $C^0(x, y)$ which describes the function $f(x, y)$. We use the notation com_b to refer to a statistically binding commitment scheme, com_h to refer to a statistically hiding commitment scheme, and com_c to refer to a commitment scheme which is only computationally binding and hiding. See Section 4 for our precise choice of these protocols.

The protocol itself is quite elaborate, but, as demonstrated in Section 5, it can be implemented quite efficiently.

0. Circuit construction: The parties replace C^0, in which P_2 has n input wires, with a circuit C in which P_2 has ℓ input wires, where $\ell = \max(4n, 8s_2)$. The only difference between the circuits is that each original input wire of P_2 in C^0 is replaced with an internal value which is computed as the exclusive-or of a random subset of the ℓ input wires of C. (Given an input to the original circuit, P_2 should therefore choose a random input to the new circuit, subject to the constraint that the internal values are equal to the original input values.) The exact construction is presented in Section 5.2 of [13]. (In order to avoid unnecessary extra gates in the circuit segment that computes the original input wires as a function of the new input wires, we designed the exact wiring using a variant of structured Gaussian elimination.)

   We let the new input wires of P_2 be given by $\hat{y} \leftarrow \hat{y}_1, \ldots, \hat{y}_\ell$.
1. Commitment construction: P_1 *constructs the circuits and commits to them, as follows:*[4]
   (a) P_1 constructs s_1 independent copies of a garbled circuit of C, denoted $GC_1, \ldots, GC_{s_1}$.
   (b) P_1 commits to the garbled values of the wires corresponding to P_2's input to each circuit. That is, for every input wire i corresponding to an input bit of P_2, and for every circuit GC_r, P_1 computes the ordered pair

$$(c^0_{i,r}, c^1_{i,r}) \leftarrow (\mathrm{com}_c(k^0_{i,r}), \mathrm{com}_c(k^1_{i,r})),$$

   where $k^b_{i,r}$ is the garbled value associated with b on input wire i in circuit GC_r. We let $(dc^0_{i,r}, dc^1_{i,r})$ denote the associated decommitment values.

[4] In [13] this commitment is done with a perfectly binding commitment scheme, however one which is computationally binding will suffice to guarantee security.

(c) P_1 computes commitment-sets for the garbled values that correspond to its own inputs to the circuits. That is, for every wire i that corresponds to an input bit of P_1, it generates s_1 pairs of commitment sets $\{W_{i,j}, W'_{i,j}\}_{j=1}^{s_1}$, in the following way:
Denote by $k_{i,r}^b$ the garbled value that was assigned by P_1 to the value $b \in \{0,1\}$ of wire i in GC_r. Then, P_1 chooses $b \leftarrow \{0,1\}$ and computes

$$W_{i,j} \leftarrow \langle \mathrm{com}_c(b), \mathrm{com}_c(k_{i,1}^b), \ldots, com_c(k_{i,s_1}^b)\rangle,$$
$$W'_{i,j} \leftarrow \langle \mathrm{com}_c(1-b), \mathrm{com}_c(k_{i,1}^{1-b}), \ldots, \mathrm{com}_c(k_{i,s_1}^{1-b})\rangle.$$

There are a total of $n \cdot s_1$ commitment-sets (s_1 per input wire). We divide them into s_1 *supersets*, where superset S_j is defined to be the set containing the jth commitment set for all wires. Namely, it is defined as $S_j = \{(W_{k,j}, W'_{k,j})\}_{k=1}^n$.

2. Oblivious transfers: For every input bit of P_2, parties P_1 and P_2 run a 1-out-of-2 oblivious transfer protocol in which P_2 receives the garbled values for the wires that correspond to its input bit (in every circuit).
Let $i_1, \ldots, i_w$ be the input wires that correspond to P_2's input, then, for every $j = 1, \ldots, w$, parties P_1 and P_2 run a 1-out-of-2 OT protocol in which:
(a) P_1's input is the pair of vectors $[dc_{i_j,1}^0, \ldots, dc_{i_j,s_1}^0]$, and $[dc_{i_j,1}^1, \ldots, dc_{i_j,s_1}^1]$.
(b) P_2's input are the bits $\hat{y}_j$, and its output should be $[dc_{i_j,1}^{\hat{y}_j}, \ldots, dc_{i_j,s_1}^{\hat{y}_j}]$.
3. Send circuits and commitments: P_1 sends to P_2 the garbled circuits, as well as all of the commitments that it prepared above.
4. Prepare challenge strings:[5]
(a) P_2 chooses a random string $\rho_2 \leftarrow \{0,1\}^{s_1}$ and sends $\mathrm{com}_h(\rho_2)$ to P_1.
(b) P_1 chooses a random string $\rho_1 \in \{0,1\}^{s_1}$ and sends $\mathrm{com}_b(\rho_1)$ to P_2.
(c) P_2 decommits, revealing ρ_2.
(d) P_1 decommits, revealing ρ_1.
(e) P_1 andP_2 set $\rho \leftarrow \rho_1 \oplus \rho_2$.
The above steps are run a second time, defining an additional string ρ'.
5. Decommitment phase for check-circuits: We refer to the circuits for which the corresponding bit in ρ is 1 as check-circuits, and we refer to the other circuits as evaluation-circuits. Likewise, if the jth bit of ρ' equals 1, then all commitments sets in superset $S_j = \{(W_{i,j}, W'_{i,j})\}_{i=1}^n$ are referred to as check-sets; otherwise, they are referred to as evaluation-sets.
For every *check-circuit* GC_r, party P_1 operates in the following way:
(a) For every input wire i corresponding to an input bit of P_2, party P_1 decommits to the pair $(c_{i,r}^0, c_{i,r}^1)$.
(b) For every input wire i corresponding to an input bit of P_1, party P_1 decommits to the appropriate values in the *check-sets* $\{W_{i,j}, W'_{i,j}\}$.
For every pair of *check-sets* $(W_{i,j}, W'_{i,j})$, party P_1 decommits to the first value in each set i.e., to the value that is supposed to be a commitment to the indicator bit, com(0) or com(1)).

[5] In [13] it is proposed to use perfectly binding and computationally hiding commitments here, but statistically binding and computationally hiding commitments actually suffice.

6. Decommitment phase for P_1's input in evaluation-circuits: P_1 decommits to the garbled values that correspond to its inputs in the evaluation-circuits.
7. Correctness and consistency checks: Player P_2 performs the following checks; if any of them fails it aborts.
   (a) *Checking correctness of the check-circuits:* P_2 verifies that each check-circuit GC_i is a garbled version of C.
   (b) *Verifying P_2's input in the check-circuits:* P_2 verifies that P_1's decommitments to the wires corresponding to P_2's input values in the check-circuits are correct, and agree with the logical values of these wires (the indicator bits). P_2 also checks that the inputs it learned in the oblivious transfer stage for the check-circuits correspond to its actual input.
   (c) *Checking P_1's input to evaluation-circuits:* Finally, P_2 verifies that for every input wire i of P_1 the following two properties hold:
      i. In every evaluation-set, P_1 chooses one of the two sets and decommitted to all the commitments in it which correspond to evaluation-circuits.
      ii. For every evaluation-circuit, all of the commitments that P_1 opened in evaluation-sets commit to the same garbled value.
8. Circuit evaluation: If any of the above checks fails, P_2 aborts and outputs $\perp$. Otherwise, P_2 evaluates the evaluation circuits (in the same way as for the semi-honest protocol of Yao). It might be that in certain circuits the garbled values provided for P_1's inputs, or the garbled values learned by P_2 in the OT stage, do not match the tables and so decryption of the circuit fails. In this case P_2 also aborts and outputs $\perp$. Otherwise, P_2 takes the output that appears in most circuits, and outputs it.

3.2 The Statistical Security Parameters

The protocol uses two statistical security parameters, s_1 and s_2. The parameter s_1 is mainly used to prevent P_1 from changing the circuit that is evaluated, or providing inconsistent inputs to different copies of the circuit. The protocol requires P_1 to provide s_1 copies of the garbled circuit, and provide $(s_1)^2$ commitments for each of its input bits. The security proof in [13] shows that a corrupt P_1 can cheat with a success probability that is exponentially small in s_1. The original proof in [13] bounds the cheating probability at $2^{-s_1/17}$, which would require a large value of s_1 in order to provide a meaningful security guarantee. We conjecture that a finer analysis can provide a bound of $2^{-s_1/4}$, and in the full version of this paper we intend to prove this; this conjecture is based on an analysis of a similar problem that was shown in [10]. A bound of $2^{-s_1/4}$ would mean that a relatively moderate value of s_1 can be used.[6]

[6] The experiments in Section 5 assume a bound of $2^{-s_1/4}$. The overhead of different parts of the protocol is either linear or quadratic in s_1. If we end up using a worse bound of $2^{-s_1/c}$, where $4 < c \leq 17$, the timings in the experiments will be increased by factor in the range $c/4$ to $(c/4)^2$.

The parameter s_2 is used to prevent a different attack by P_1, in which it provides corrupt values to certain inputs of the oblivious transfer protocol and then uses P_2's reaction to these values to deduce information about P_2's inputs (see [13] for details). It was shown that setting the number of new inputs to be $\ell = max(4n, 8s_2)$ bounds the success probability of this type of attack by 2^{-s_2}. The values of s_1 and s_2 should therefore be chosen subject to the constraint that the total success probability of a cheating attempt, $\max(2^{-s_1/4}, 2^{-s_2})$, is acceptable. Therefore, one should set $s_1 = 4s_2$.

3.3 Optimizing the Protocol Components

The protocol uses many components, which affect its overall overhead. These include the encryption scheme, the commitment schemes, and oblivious transfer. Much of our work was concerned with optimizing these components, in order to improve the performance of the entire protocol. We describe in the next section the different optimizations that we applied to the different components.

4 Subprotocols

To implement the above protocol requires us to define a number of sub-protocols: various commitment schemes, OT protocols and encryption schemes. In what follows we select the most efficient schemes we know of, in both the random oracle model (ROM) and the standard model. We assume that the concrete computational security parameter (as opposed to the statistical security parameter) is given by t. By this we mean that we select primitives which have security equivalent to t bits of block cipher security. Thus we first select an elliptic curve E of prime order $q \approx 2^{2t}$, and a symmetric cryptographic function with a t-bit key.

Elliptic curve. We let $\langle P \rangle = \langle Q \rangle = E$, an elliptic curve of prime order $q \approx 2^{2t}$, where no party knows the discrete logarithm of Q with respect to P.

Symmetric cryptographic function. The function that will be used for symmetric key cryptography is defined as a key derivation function $\mathrm{KDF}(m, l)$, which takes an input string m and outputs a bit string of length l. We use the KDF defined in ANSI X9.63, which is the standard KDF to use in the elliptic curve community [19]. It is essentially implemented as encryption in CTR mode where the encryption function is replaced by the SHA-1 hash function.

4.1 Encryption Scheme for Garbled Circuits

The encryption scheme $E^s_{k_1,k_2}(m)$ used to encrypt the values in the Yao circuit is defined by the algorithms in Figure 1. We assume that $k_i \in \{0,1\}^t$. The ROM version is secure on the assumption that the function KDF is modelled as a random oracle, whilst the standard model scheme is secure on the assumption that $\mathrm{KDF}(k \| s, l)$ is a pseudo-random function, when considered as a function on s keyed by the key k. We remark that the encryption is secure as long as

the string s is used only once for any choice of key k. Note that the non-ROM version requires two invocations of the KDF, since we do not know how to analyze the security of a pseudo-random function if part of its key is known to an adversary (namely, if we use $\mathrm{KDF}(k_1\|k_2\|s,|m|)$, where KDF is modeled as a pseudo-random function, k_2 is secret and k_1 is known to an adversary, we cannot argue that the output is pseudo-random).

Input: Keys k_1, k_2 of length t, and a string s. For encryption an l-bit message m in also given. For decryption, an l-bit ciphertext c is given.

ROM Version

Encryption $E^s_{k_1,k_2}(m)$

1. $k \leftarrow \mathrm{KDF}(k_1\|k_2\|s,|m|)$.
2. $c \leftarrow k \oplus m$.

Decryption

1. $k \leftarrow \mathrm{KDF}(k_1\|k_2\|s,|m|)$.
2. $m \leftarrow k \oplus c$.
3. Return m.

Non-ROM Version

Encryption $E^s_{k_1,k_2}(m)$

1. $k \leftarrow \mathrm{KDF}(k_1\|s,|m|)$.
2. $k' \leftarrow \mathrm{KDF}(k_2\|s,|m|)$.
3. $c \leftarrow m \oplus k \oplus k'$

Decryption

1. $k \leftarrow \mathrm{KDF}(k_1\|s,|c|)$.
2. $k' \leftarrow \mathrm{KDF}(k_2\|s,|c|)$.
3. $m \leftarrow c \oplus k \oplus k'$.
4. Return m.

Fig. 1. ROM and non-ROM encryption algorithms for the Yao circuits

4.2 Commitment Schemes

Recall we have three types of commitment schemes; statistically binding, statistically hiding and computationally binding/hiding, to commit to a value $m \in \{0,1\}^t$. (Note that the elliptic curve E is of order $q \approx 2^{2t}$ and so we can view m as a number in $\mathbb{Z}_q$ if desired.)

A Statistically Binding Commitment : $\mathbf{com_b}(m)$
We define the statistically binding commitment scheme as in Figure 2. The random oracle model based scheme is statistically binding, since to break the binding property we need to find collisions in the hash function H. Since H is modelled as a random oracle, the probability of any adversary finding a collision given a polynomial number of points of H is negligible, even if it is computationally unbounded. The scheme is also computationally hiding by the fact that H is modelled as a random oracle (in fact, it's even statistically hiding if the adversary is limited to a polynomial number of points of H). The non-ROM scheme is statistically binding because P and c_1 fully determine r, which together with Q and c_2 in turn fully determine m. The fact that it is computationally hiding follows directly from the DDH assumption over the elliptic curve used.

ROM Version

H is a hash function modeled as a random oracle.

Commitment $\text{com}_b(m)$

1. $r \leftarrow \{0,1\}^t$.
2. $c \leftarrow H(m\|r)$.
3. Return c.

Decommitment

1. Reveal m and r.
2. Check if $c = H(m\|r)$.
3. Return m.

Non-ROM Version

P and Q are elements on an elliptic curve, as described above.

Commitment $\text{com}_b(m)$

1. $r \leftarrow \mathbb{Z}_q$.
2. $c_1 \leftarrow [r]P$.
3. $c_2 \leftarrow [r][m]Q$.
4. Return (c_1, c_2).

Decommitment

1. Reveal m and r.
2. Check if $c_1 = [r]P$.
3. Check if $c_2 = [r][m]Q$.
4. Return m.

Fig. 2. ROM and non-ROM statistically binding commitment schemes

The Statistically Hiding Commitment : $\text{com}_h(m)$

For the statistically hiding commitment scheme we use the Pederson commitment [18]:

$$\text{com}_h(m) \leftarrow [r]P + [m]Q$$

where r is a random number of size q and we treat m as an integer modulo q. Note that $0 \leq m < 2^t < q < 2^{2t}$. Decommitment is performed by revealing r and m, and then verifying the commitment is valid. This is actually a perfectly hiding commitment (since given $\text{com}_h(m)$ there exists, for any possible value of m', a corresponding value r' for which $\text{com}_h(m) = [r']P + [m']Q$) and so in particular the commitment is also statistically hiding. That the commitment is computationally binding follows from the fact that any adversary who can break the binding property can determine the discrete logarithm of Q with respect to P.

A Computational Commitment Scheme : $\text{com}_c(m)$

We use the ROM version of the statistically binding commitment scheme in Figure 2 for both the ROM and non-ROM commitments here. This is clearly suitable in the ROM. Regarding the non-ROM case, this scheme is computationally binding on the assumption that H is collision-resistant. Furthermore, it is computationally hiding when $H(m\|r)$ is modelled as a PRF with key r and message m. We remark that when m is large, this latter assumption clearly does not hold for typical hash functions based on the Merkle-Damgård paradigm (where given $H(k\|m)$ one can easily compute $H(k\|m\|m')$ for some m'). However, it is reasonable when m fits into a single iteration of the underlying compression function (as is the case here where $m \in \{0,1\}^t$ and t is a computational security parameter which we set to the value $t = 128$.).

4.3 Oblivious Transfer

Recall in our main protocol we need to perform $w = \max(4n, 8s_2)$ 1-out-of-2 oblivious transfers in Stage 2. We batch these up so as to perform all the OT's in a single batch. The OT's need to be performed in a manner which has a simulation based proof of security against malicious adversaries, hence the simple protocols of [17,1,12] are not suitable for our purposes (the simulation based proof is needed in order to be able to use a composition of the OT protocol in our protocol, see [6]). We therefore use a modification of the batched version of the protocol of Hazay and Lindell [10], which we now describe in the elliptic curve setting. (We note that this protocol has a simulation based proof of security in the standard model, without any usage of a random oracle.)

We assume that P_1's input is two vectors of values

$$[x_1^0, \ldots, x_w^0] \text{ and } [x_1^1, \ldots, x_w^1],$$

where $|x_j^0| = |x_j^1|$. Party P_2 has as input the bits $i_1, \ldots, i_w$ and wishes to obtain the vector $[x_1^{i_1}, \ldots, x_w^{i_w}]$.

We assume two zero-knowledge proofs-of-knowledge protocols which we shall describe in Appendix A. The first, $DL([x]P; x)$, proves, in zero-knowledge, knowledge of the discrete logarithm x of $[x]P$; the second, $DH(P, [a]P, [b]P, [ab]P)$, proves that the tuple P,$[a]P$,$[b]P$,$[ab]P$ is a Diffie–Hellman tuple.

The protocol follows. The main things to notice are that the zero-knowledge proofs of knowledge are performed only once, regardless of the number of items to be transfered, and that protocol is composed of only two rounds (in addition to the rounds needed by the zero-knowledge proofs).

1. P_2 chooses $\alpha_0, \alpha_1 \in \mathbb{Z}_q$ and computes $Q_0 \leftarrow [\alpha_0]P$ and $Q_1 \leftarrow [\alpha_1]P$, it then executes the protocol $DL(Q_0; \alpha_0)$ with party P_1.
2. For $j = 1, \ldots, w$ party P_2 chooses $r_j \in \mathbb{Z}_q$ and computes $U_j \leftarrow [r_j]P$, $V_{0,j} \leftarrow [r_j]Q_0 + [i_j]P$, $V_{1,j} \leftarrow [r_j]Q_1 + [i_j]P$. These values are then sent to P_1.
3. P_1 chooses $\rho_j \in \mathbb{Z}_q$, for $j = 1, \ldots, w$ and sends them to P_2.
4. Both parties then locally compute

$$U \leftarrow \sum_{j=1}^{w} [\rho_j] U_j, \quad V \leftarrow \sum_{j=1}^{w} [\rho_j](V_{0,j} - V_{1,j}).$$

   Party P_2 executes the protocol $DH(P, Q_0 - Q_1, U, V)$ with party P_1.
5. For $j = 1, \ldots, w$ P_1 then performs the following steps:
   (a) Select $R_{0,j}, R_{1,j} \in \langle P \rangle$ at random.
   (b) Select $s_{0,j}, t_{0,j}, s_{1,j}, t_{1,j} \in \mathbb{Z}_q$.
   (c) Set $e_{0,j} \leftarrow (W_{0,j}, Z_{0,j}, y_{0,j})$ where

$$\begin{aligned} W_{0,j} &\leftarrow [s_{0,j}]U + [t_{0,j}]P, \\ Z_{0,j} &\leftarrow [s_{0,j}]V_0 + [t_{0,j}]Q_0 + R_{0,j}, \\ y_{0,j} &\leftarrow x_j^0 \oplus \mathrm{KDF}(R_{0,j}, |x_j^0|). \end{aligned}$$

(d) Set $e_{1,j} \leftarrow (W_{1,j}, Z_{1,j}, y_{1,j})$ where

$$\begin{aligned} W_{1,j} &\leftarrow [s_{1,j}]U + [t_{1,j}]P, \\ Z_{1,j} &\leftarrow [s_{1,j}](V_1 - P) + [t_{1,j}]Q_1 + R_{1,j}, \\ y_{1,j} &\leftarrow x_j^1 \oplus \mathrm{KDF}(R_{1,j}, |x_j^1|). \end{aligned}$$

The values $(e_{0,j}, e_{1,j})$ are then sent to P_2 for each value of j.

6. For $j = 1, \ldots, w$, party P_2 then computes

$$R \leftarrow Z_{i_j,j} - [\alpha_{i_j}]W_{i_j,j}$$

and outputs

$$x_j^{i_j} \leftarrow y_{i_j,j} \oplus \mathrm{KDF}(R, |x_j^{i_j}|).$$

For each index in the vector of inputs, the protocol requires P_1 to perform 10 multiplications, and P_2 to perform 8 multiplications. (This is without considering the zero-knowledge proofs, which are performed once in the protocol.) The security of the above scheme is fully proven in [10], with the only exception that here a KDF is used to derive a random string in order to mask (i.e., encrypt) the x_j^0 and x_j^1 values (in [10] it is assumed that x_j^0 and x_j^1 can be mapped into points in the Diffie-Hellman group). The use of a KDF for this purpose was proven secure in the context of hashed ElGamal in [22], on the assumption that KDF is chosen from a family of hash functions which are entropy smoothing.

5 Timings

In our implementation we selected $t = 128$ as the security parameter. As a result, we chose the KDF to be implemented by SHA-256, and as the elliptic curve E we selected the curve *secp256r1* from the SECG standard [20].

We performed a set of experiments which examined the system using a circuit which evaluates the function $x > y$ for inputs x and y of $n = 16$ bits in length. The standard circuit (using simple 2-to-1 gates) for this problem consists of 61 2-to-1 gates and 93 internal wires. We optimized this circuit by replacing it with a circuit consisting of 48 internal wires and fifteen 3-to-1 gates and one 2-to-1 gate. We only looked at the case of P_2 obtaining the result, the extension to the first party obtaining the result is standard and requires an extension to the circuit to be made, for which similar optimizations can be made.

The size of the modified circuit: Step 0 of the protocol replaces the circuit with a different one which has $\max(4n, 8s_2)$ input wires. The statistical security parameter s_2 therefore affects the size of the circuit, both in terms of the number of wires and the number of gates. When $n < 2s_2$, as in our experiments, we have $8s_2$ new input wires. Each original input wire is replaced with the exclusive-or of about $4s_2$ input wires, which can be computed using $4s_2 - 1$ gates. The circuit therefore grows by about $4ns_2$ gates, which in our case translate to 2560 gates for $s_2 = 40$, and 3840 gates for $s_2 = 60$. We managed to optimize this construction

by using a variant of structured Gaussian elimination in order to reuse gates. As a result, for the case of $s_2 = 40$, the augmented circuit produced in Stage 0 has over one thousand gates and over one thousand five hundred internal wires. If s_2 is increased to 60 then the augmented circuit now has over one thousand five hundred gates and over two thousand internal wires. The exact increase in size depends on the random choices made in Stage 0, but the above values are indicative.

Implementation: The program was implemented in C++ using standard libraries; the elliptic curve routines made use of specially written assembly functions to perform the arithmetic instructions. On the machine that was used for the experiments, and the curve we were using, the software needed 3.9 milliseconds for a basic multiplication, 1.2 milliseconds to multiply the fixed generator, and 5.1 milliseconds in order to compute $(aP + bQ)$ (using a variant of the method described in Algorithm 14.88 of [16]).

The input to the program was a circuit represented by a text file, each line of the text file represented a gate. For example the line

```
2 1 0 16 32 0100
```

represents a 2-to-1 gate which has input wires numbered 0 and 16 and produces the output wire 32. The value of the gate is given by the string which follows. The above example implements a two-bit "less than" gate, namely it will output a 1 on wire 32 only if $w_0 < w_{16}$, i.e. the value of wire 0 is zero and the value of wire 16 is one.

Experiments: We performed a set of experiments with different values of the statistical security parameters s_1 and s_2, and using both the ROM and standard model versions of the protocol. The run times, in seconds, are presented in Table 1, and are reported for each step of the protocol. Timings are performed using the standard Linux system timing facilities, and are as such only indicative. The wall time is measured using the standard *time* function and the system and user times are measured using the *getrusage* function. The wall time represents the elapsed wall clock time in running the program, the user time represents the amount of time each party actually performed some computation, whereas the syst time represents the time spent by each party in system routines (for example transmitting data, or writing to disk, etc.). All timings were performed on an Intel Core 2 6420 running at 2.13 GHZ with 4096 KB of cache and 2 GB of RAM and are given in seconds.

Basic observations: The computation is not instantaneous but overall the run time is quite reasonable (the overall run time is about 2-3 minutes for a security parameter $s_1 = 160$). The run time is affected, of course, by the fact that 160 copies of the circuit are being used in the computation (compared to a protocol secure against semi-honest adversaries, which uses only a single copy of the circuit), and the fact that each circuit is much larger than its original form (in the experiment more than 1000 gates are added to the circuit in Step 0, where the original circuit consisted of less than 20 gates).

Oblivious transfers: It is a little surprising that Step 2, which includes the oblivious transfers, is not the main bottleneck of the protocol. This is true even though we implemented an OT protocol which is secure against malicious adversaries according to a full simulation definition.

Preprocessing: About half of the run time is consumed by Step 1, where P_1 prepares the circuits and the commitments. This step can be run offline, before the inputs are known, reducing the online run time by about 50%.

Scaling: Increasing s_1 by a factor of c_1 increases by a factor of c_1^2 the number of commitments generated by P_1 in Step 1, and increases the number of circuits by c_1. Increasing s_2 by a factor of c_2 increases the size of the modified part of the circuit (which is the bulk of the circuit in our experiments) by a factor of c_2, and therefore the total size of the circuits is increased by a factor of $c_1 c_2$. In the experiments, we increased both s_1 and s_2 by a factor of 1.5 (from 40 to 60, and from 160 to 240, respectively). We therefore expected the overhead to increase by a factor of 2.25. The actual measurements showed an increase by a factor slightly larger than 2.

We did not conduct experiments with circuits of different sizes. When all other parameters are fixed, we expect the run time to be linear in the size of the *modified circuit* (after the modifications done in Step 0). We can estimate the size of the modified circuit as follows: If P_2 has n input wires in the original circuit, then the modified circuit is expected to have about $\frac{n}{2}\max(4n, 8s_2)$ more gates. (Applying structured Gaussian elimination can enable us to reuse gates and minimize the size of the modified circuit.)

Performance in the ROM and in the standard model: What is interesting about the timings is that there is very little difference between the timings in the ROM and those in the standard model. In Step 1 the ROM version is more efficient simply due to the slightly more efficient encryption scheme used.[7] Given the large number of encryptions needed to produce the garbled circuit this translates into a small advantage for the ROM version compared to the standard-model implementation. For a similar reason one obtains a performance improvement in the ROM in Step 7 in which the circuit is evaluated by P_2. The decrease in performance of the ROM compared to the standard model in Step 3 we cannot explain, but it is likely to be caused by experimental error.

In viewing the timings it should be born in mind that the main place that the random oracle model is used is in the oblivious transfers in Step 2. At this point we use the ROM to reduce the round complexity of the two required zero-knowledge proofs (see Appendix A for details of this). However, these two

[7] The KDF is invoked in the standard model protocol about twice as many times as in the ROM protocol (since the encryption function in the standard model calls the KDF twice). The increase in the run time of Step 1 when changing the ROM implementation to the standard-model implementation (for $s_1 = 160$) is from 60sec to 67sec. We therefore estimate that the circuit construction (Step 1(a)) takes about 7 seconds in the ROM protocol and 14 seconds in the standard model protocol.

proofs are only used once in the whole run of the protocol as we have batched the oblivious transfers, and therefore the run time of Step 2 is about the same in both the ROM and the standard model protocols.

What is surprising about the fact that the standard model is comparable in performance to the ROM is that for simpler cryptographic functionalities, such as encryption or signature schemes, the performance of the best ROM based scheme is often twice as fast as the best known standard model scheme.

6 Future Work

An obvious task is to develop the current implementation into a complete system for secure computation. In particular, the system should include a front end that will enable users to provide a high-level specification of the function that they want to compute, and specify the different security parameters that shall be used. A natural approach for this task would be to modify the FairPlay compiler [15] to support our implementation.

Table 1. Run times of our experiments

Run Times in the Random Oracle Model

	Step								
Time	1	2	3	4	5	6	7	8	Total
P_1, $s_1 = 160$, $s_2 = 40$									
Wall	74	20	24	0	7	10	0	0	135
User	60	17	12	0	3	4	0	0	
Syst	16	2	3	0	0	0	0	0	
P_2, $s_1 = 160$, $s_2 = 40$									
Wall	74	20	24	0	8	9	35	1	171
User	0	8	14	0	8	7	29	1	
Syst	0	0	10	0	2	4	8	0	
P_1, $s_1 = 240$, $s_2 = 60$									
Wall	159	34	51	0	19	13	0	0	276
User	123	30	24	0	11	6	0	0	
Syst	35	2	9	0	1	0	0	0	
P_2, $s_1 = 240$, $s_2 = 60$									
Wall	159	34	51	0	19	13	78	3	358
User	0	12	28	0	17	10	61	2	
Syst	0	0	22	0	7	5	18	0	

Run Times in the standard Model

	Step								
Time	1	2	3	4	5	6	7	8	Total
P_1, $s_1 = 160$, $s_2 = 40$									
Wall	84	20	24	0	7	7	0	0	142
User	67	18	10	0	5	3	0	0	
Syst	15	0	5	0	0	0	0	0	
P_2, $s_1 = 160$, $s_2 = 40$									
Wall	84	20	24	0	7	7	40	2	184
User	0	10	13	0	7	5	32	4	
Syst	0	0	11	0	1	3	8	2	
P_1, $s_1 = 240$, $s_2 = 60$									
Wall	181	35	45	0	18	12	0	0	291
User	145	30	24	0	8	8	0	0	
Syst	35	0	7	0	1	2	0	0	
P_2, $s_1 = 240$, $s_2 = 60$									
Wall	181	35	45	0	18	12	87	5	362
User	0	12	23	0	15	9	70	7	
Syst	0	0	21	0	4	3	20	0	

The performance of the system is greatly affected by the circuit modification in Step 0 of the protocol, which increases the number of inputs and the size of the circuit. We implemented this step according to the randomized construction in [13]. Another option is to use a linear error-correction code for defining the relation between the original and new input wires of the circuit. (A careful examination of

the proof in [13] shows that this is sufficient.) We need an $[N, k, d]$ linear binary code which encodes k bit words into N bit words with a distance of $d = s_2$ (say, $d = 40$). The parameter k corresponds to the number of original input wires of P_2, while N corresponds to the number of new input wires. The code should satisfy two criteria: (1) the rate k/N should be as high as possible, to keep the number of new input wires close to the number of original input wires, and (2) the block length k should be minimized, to enable the code to be applied (and the rate k/N to be achieved) even if P_2's input is relatively short.

References

1. Aiello, B., Ishai, Y., Reingold, O.: Priced Oblivious Transfer: How to Sell Digital Goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001)
2. Aggarwal, G., Mishra, N., Pinkas, B.: Secure Computation of the k-th Ranked Element. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 40–55. Springer, Heidelberg (2004)
3. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP – A System for Secure Multi-Party Computation, manuscript (2008)
4. Bogetoft, P., Christensen, D.L., Dåmgard, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Multiparty Computation Goes Live, Cryptology ePrint Archive 2008/068 (2008)
5. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J.: A practical implementation of secure auctions based on multiparty integer computation. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006)
6. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13(1), 143–202 (2000)
7. Chaum, D., Pederson, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
8. Goldreich, O.: Foundations of Cryptography: Volume 2 – Basic Applications. Cambridge Univ. Press, Cambridge (2004)
9. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In: 19th STOC, pp. 218–229 (1987)
10. Hazay, C., Lindell, Y.: Oblivious transfer, polynomial evaluation and set intersection. Manuscript (2008)
11. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
12. Kalai, Y.T.: Smooth Projective Hashing and Two-Message Oblivious Transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer, Heidelberg (2005)
13. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
14. Malkhi, D., Franklin, M.K.: Efficiency tradeoffs for malicious two-party computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006)

15. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay — a secure two-party computation system. In: Proc. of 13th USENIX Security Symposium (2004)
16. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
17. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: 12th SODA, pp. 448–457 (2001)
18. Pederson, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
19. Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography, `http://www.secg.org/download/aid-385/sec1_final.pdf`
20. SECG. Standards for Efficient Cryptography, SEC 2: Recommended elliptic curve domain parameters, `http://www.secg.org`
21. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (1990)
22. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs. Manuscript (2004)
23. Woodruff, D.: Revisiting the Efficiency of Malicious Two-Party Computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 79–96. Springer, Heidelberg (2007)
24. Yao, A.: How to generate and exchange secrets. In: 27th FOCS, pp. 162–167 (1986)

A Zero Knowledge Proofs

We now describe the zero-knowledge proof-of-knowledge protocols required in the OT protocol. In the ROM we use the standard Fiat-Shamir transform of an interactive honest-verifier Σ-protocol into a non-interactive protocol via hashing the commitment with the random oracle so as to produce the random challenge.

In the standard model we need to cope with non-honest verifiers by getting the verifier to commit to his challenge before the prover's commitment is issued. We use a highly-efficient transformation described in [10] to transform an honest-verifier Sigma protocol to a protocol that is a zero-knowledge proof of knowledge (the transformation is proven secure under the assumption that the discrete logarithm problem is hard and hence is highly suitable for proofs of Diffie-Hellman type statements).

A.1 $DL(Q; x)$

We assume a prover *Pro* who knows x and a verifier *Ver* who only knows Q and P. The two protocols, one in the ROM and one in the standard model, are presented in Fig. 3. They are based on the HVZK proof of Schnorr [21].

A.2 $DDH(P, [a]P, [b]P, [ab]P)$

We assume a prover *Pro* who knows b and a verifier *Ver* who only knows the four protocol inputs P, $Q = [a]P$, $U = [b]P$ and $V = [b]Q$. The two variants of the protocol are given in Fig. 4, both are based on the HVZK protocol from [7].

ROM Version

- *Pro* computes $k \leftarrow \mathbb{Z}_q$, $R \leftarrow [k]P$, $s \leftarrow H(R)$, $z \leftarrow xs + k$. It sends R and z to *Ver*.
- *Ver* computes $s \leftarrow H(R)$. and accepts if $[z]P = [s]Q + R$.

Non-ROM Version

- *Pro* computes $a \leftarrow \mathbb{Z}_q$, $A \leftarrow [a]P$. It sends A to *Ver*.
- *Ver* computes $s, t \leftarrow \mathbb{Z}_q$, $C \leftarrow [s]P + [t]A$. and sends C to *Pro*.
- *Pro* computes $k \leftarrow \mathbb{Z}_q$, $R \leftarrow [k]P$. and sends R to *Ver*.
- *Ver* sends s, t to *Pro*.
- *Pro* checks whether $C = [s]P + [t]A$. and sends $z \leftarrow xs + k$ and a to *Ver*.
- *Ver* accepts if $[z]P = [s]Q + R$ and $A = [a]P$.

Fig. 3. ROM and non-ROM zero-knowledge proof of knowledge of discrete logarithms

ROM Version

- *Pro* computes $r \leftarrow \mathbb{Z}_q$, $A \leftarrow [r]P$, $B \leftarrow [r]Q$, $s \leftarrow H(A\|B)$, $z \leftarrow bs + r$. and sends A, B and z to *Ver*.
- *Ver* computes $s \leftarrow H(A\|B)$ and accepts if $[z]P = [s]U + A$ and $[z]Q = [s]V + B$.

Non-ROM Version

- *Pro* computes $w \leftarrow \mathbb{Z}_q$, $W \leftarrow [w]P$ and sends V to *Ver*.
- *Ver* computes $s, t \leftarrow \mathbb{Z}_q$, $C \leftarrow [s]P + [t]A$ and sends C to *Pro*.
- *Pro* computes $r \leftarrow \mathbb{Z}_q$, $A \leftarrow [r]P$, $B \leftarrow [r]Q$ and sends A and B to *Ver*.
- *Ver* sends s, t to *Pro*.
- *Pro* checks whether $C = [s]P + [t]V$. and sends $z \leftarrow bs + r$ and w to *Ver*.
- *Ver* accepts if $[z]P = [s]U + A$, $[z]Q = [s]V + B$ and $W = [w]P$.

Fig. 4. ROM and non-ROM zero-knowledge proof of knowledge of DDH tuple

CLL: A Cryptographic Link Layer for Local Area Networks

Yves Igor Jerschow, Christian Lochert, Björn Scheuermann, and Martin Mauve

Institute of Computer Science, Heinrich Heine University, Düsseldorf, Germany
{jerschow,lochert,scheuermann,mauve}@cs.uni-duesseldorf.de

Abstract. Ethernet and IP form the basis of the vast majority of LAN installations. But these protocols do not provide comprehensive security mechanisms, and thus give way for a plethora of attack scenarios. In this paper, we introduce a layer 2/3 security extension for LANs, the *Cryptographic Link Layer (CLL)*. CLL provides authentication and confidentiality to the hosts in the LAN by safeguarding all layer 2 traffic including ARP and DHCP handshakes. It is transparent to existing protocol implementations, especially to the ARP module and to DHCP clients and servers. Beyond fending off external attackers, CLL also protects from malicious behavior of authenticated clients. We discuss the CLL protocol, motivate the underlying design decisions, and finally present implementations of CLL for both Windows and Linux. Their performance is demonstrated through real-world measurement results.

1 Introduction

Ethernet and the Internet Protocol (IP) are the main building blocks for the vast majority of modern Local Area Networks (LANs). However, these protocols, and thus virtually all installed LANs, do not provide comprehensive security mechanisms. Hence, malicious local users are potentially able to eavesdrop, to inject or modify information, or to take on fake identities.

One especially critical component is the *Address Resolution Protocol (ARP)* [20]. It performs the task of coupling the network layer with the link layer by resolving IP addresses into the corresponding MAC addresses. However, ARP lacks an authentication mechanism, making it vulnerable to different types of attacks. This constitutes a severe threat in every LAN that is accessible to not fully trustworthy users. By emitting ARP messages with wrong IP/MAC mappings—commonly referred to as *ARP spoofing*—a malicious user can *impersonate* other hosts, intercept and modify foreign IP traffic by becoming a *Man in the Middle (MiM)*, or mount a *Denial of Service (DoS)* attack against other hosts. Using freely available tools, e. g. [18, 9], ARP spoofing can be easily performed even by users without deeper knowledge of the underlying protocols.

Preventing ARP attacks in the case of dynamic IP addresses requires to take also the *Dynamic Host Configuration Protocol (DHCP)* [7] into account. It is employed in almost every LAN to automatically assign IP addresses and configuration parameters. It does not provide an authentication mechanism either

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 21–38, 2008.

and thus can also become the target of various attacks. By setting up a rogue DHCP server and announcing forged IP addresses for the default gateway or the DNS server, an adversary is able to run a MiM or DoS attack against clients requesting an IP address via DHCP. Furthermore, the legitimate DHCP server is also vulnerable. In a *DHCP starvation attack* the adversary takes on many different client identities (usually MAC addresses) and requests each time a new IP address, until the server's address pool gets exhausted. Thereby the attacker can prevent new clients from acquiring a valid IP configuration.

Since modern operating systems enable the injection of raw Ethernet packets containing arbitrary MAC and IP addresses in their headers even in user mode, there exists no external barrier which would impede address fraud. The outlined attack scenarios are covered in more detail, e. g., in [1, 5, 23].

In this paper, we tackle the challenge of securing the communication in local area networks, including ARP and DHCP. We introduce a comprehensive layer 2/3 security protocol—the *Cryptographic Link Layer (CLL)*. It provides authentication and confidentiality between neighboring hosts in Ethernet LANs. Each machine gets identified by its IP/MAC address pair. Beyond safeguarding ARP and DHCP, CLL protects arbitrary layer 2 traffic, especially all encapsulated IP packets. We propose to employ CLL, e. g., in enterprise and campus networks being often accessed by frequently changing, not fully trustworthy users as well as in all kinds of publicly accessible LANs (like Internet cafés or Wi-Fi hotspots). Note that CLL does not affect the operation of higher layer security protocols.

Beginning with an ARP request, CLL applies public key cryptography to perform an initial handshake between two hosts with the aim to establish a security association. The two hosts prove their identity to each other and exchange keying material. Hereupon, secured IP data packets may be sent.

We have implemented and evaluated CLL on both Windows and Linux. In typical LANs running at 100 Mbit/s, our implementation operates at full wire-speed, thus securing the network without compromising the throughput. To ease the migration procedure, CLL-enabled machines can be configured to interoperate with ordinary, unsecured hosts. We make our CLL implementation available for free download including the sources, and complement it with a toolkit for key and certificate management [12].

The remainder of this paper is organized as follows. In the next section, we review previous approaches on securing ARP, DHCP, and the link layer. Section 3 sketches the architecture of CLL, before Section 4 justifies the underlying cryptographic design decisions. In Sections 5 and 6 we detail the operation of CLL's protocol components. Section 7 describes the implementation of CLL and evaluates its performance. Finally, we conclude the paper with a summary in Section 8.

2 Related Work

Above the link layer, there already exist well-proven security protocols which provide authentication and confidentiality by means of cryptography. *SSH* [24] and *SSL / TLS* [6] operate at the application level or directly below it. At the

network layer, *IPsec* [13] can protect IP datagrams being exchanged between two end-points. However, IPsec does not authenticate the IP address of the communicating party. This enables an authorized IPsec user to impersonate the IP address of another host that is temporarily switched off or knocked out by a DoS attack. While SSH, SSL/TLS, and IPsec cannot protect from attacks on ARP and DHCP, the encryption performed by these protocols will still prevent the disclosure of sensitive data. An attacker would have to content himself with the power of rendering his victims unable to communicate.

Reviewing the attempts to cope with the insecurity of ARP, there exist two main directions. One is to detect the bulk of ARP attacks by means of a specialized Intrusion Detection System (IDS) like *Antidote* [2] or *ArpWatch* [3] and to warn the user or network administrator in time. Such tools monitor all incoming ARP messages and trigger an alarm, e. g., on observing an abnormally high number of ARP replies or a changed IP/MAC mapping. However, these ARP watchdogs cannot provide full protection against ARP attacks; in particular, they are not able to distinguish whether the MAC address in the first ARP reply is genuine or not. The other approach is to secure ARP by using cryptographic techniques. In the following, we discuss some current research taking this direction.

Gouda and Huang [10] specify a theoretical architecture with an ARP server sharing a symmetric key for message authentication with every host in the LAN. Each host periodically notifies the server about its current IP/MAC mapping and resolves the MAC addresses of its neighbors with the aid of the ARP server. However, this does not prevent an authorized machine from purposely announcing a mapping of a neighboring host's IP address to its own MAC address. In contrast, CLL authenticates all hosts based on their IP/MAC address pair. It thus also avoids ARP spoofing attempts originating from malicious, but authorized users. Furthermore, CLL does not require a central server.

In [5], Bruschi et al. introduce *Secure ARP (S-ARP)* which uses public key signatures to authenticate the ARP replies. All hosts in the LAN hold a private/public key pair and initially enroll at a central server, the *Authoritative Key Distributor (AKD)*. The AKD maintains a repository of public keys and the corresponding (static) IP addresses. Whenever a host requires a neighbor's public key to verify the signature of an ARP reply, it inquires about it from the AKD. The AKD's reply packet is digitally signed as well and the AKD's public key is preinstalled on all machines. S-ARP comes with an implementation for Linux 2.4, but it requires a kernel patch and does not support dynamically assigned IP addresses.

On the basis of S-ARP, Lootah et al. propose *Ticket-based ARP (TARP)* [16]. It foregoes a central key server and instead employs digitally signed attestations of IP/MAC mappings, so-called *tickets*. The tickets are issued by a trusted party, the *Local Ticket Agent (LTA)*. The host responding to an ARP request attaches its ticket to the ARP reply and thereby proves the validity. Since the LTA's public key is preinstalled on each host, received tickets can be verified quickly. In comparison to S-ARP, TARP requires at most one public key operation per

ARP exchange and no private key operations, and thus offers better performance. However, the authors state that an attacker is able to impersonate a host that is currently offline, by replaying its previously captured ticket. TARP has been implemented on Linux 2.6 with support for DHCP-assigned IP addresses. Note, however, that both S-ARP and TARP aim to secure only ARP, while CLL provides overall layer 2 security by safeguarding DHCP and data packets as well.

RFC 3118 [8] specifies how DHCP can be extended by an authentication mechanism. In this scheme, the DHCP server shares with each client a symmetric key. It is used to authenticate the DHCP messages. However, DHCPDISCOVER, the first message sent by the client, remains unauthenticated. Consequently, users still might be able to perform a DHCP starvation attack. This is not the case with CLL. Another drawback is that currently no DHCP implementations with RFC 3118 support are available.

Applying cryptography at the link layer is common in Wi-Fi networks. *Wi-Fi Protected Access (WPA)* and *WPA2* provide authentication and confidentiality between wireless nodes and the access point. The IEEE working group 802.1AE [11] specifies *MACsec* as the analog of WPA/WPA2 for LANs. In contrast to CLL, WPA/WPA2 and MACsec authenticate hosts based only on their MAC address. The content of ARP and DHCP control packets encapsulated in layer 2 frames is not examined. Therefore these protocols cannot protect from ARP and DHCP attacks originating from legitimate users. Moreover, we are not aware of any MACsec implementation being available at this time.

The main contribution of this paper is a novel, comprehensive approach to layer 2 security, which provides a more complete protection of the LAN than even a combination of three existing protocols (e. g., TARP, RFC 3118, and IPsec) could achieve. That is because besides eliminating the discussed shortcomings of these protocols, CLL also authenticates broadcast traffic. The tackled security problems are all related to each other—they arise from the lack of authentication at layer 2 and the link to layer 3. Thus, a comprehensive approach to solve them seems appropriate.

3 Protocol Overview

CLL is designed as a transparent filtering service between the network adapter and the IP stack. It thus operates at the border between the link and the network layer, as displayed in Figure 1. All outgoing packets including the Ethernet header are authenticated and their payload is optionally encrypted before they are handed over to the network card for transmission. Incoming packets are passed to the IP stack only after they have been successfully authenticated and—if applicable—decrypted. CLL can be enabled or disabled without modifying the other protocol stack components. For them, CLL's services are transparent. But in fact, CLL appends its cryptographic headers to outgoing packets, and puts its own ID into the EtherType field of the Ethernet header. From successfully authenticated incoming packets CLL strips off its cryptographic headers and restores the original EtherType value before passing them up. While the operation of CLL does not require any modifications to switches, routers must either

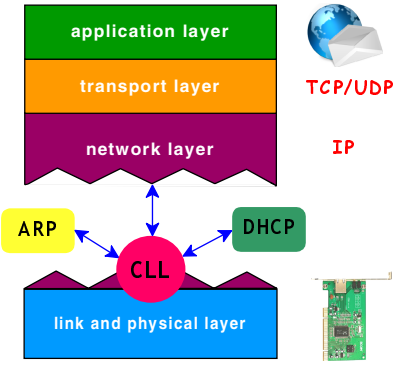


Fig. 1. CLL in the protocol stack

support CLL (and benefit from it) or exchange packets with the end systems in the standard, insecure manner.

CLL identifies hosts by their IP/MAC address pair. Each machine on the LAN holds a private/public key pair and a certificate issued by the local Certificate Authority (CA)—usually the network administrator—which establishes the binding between its public key, the MAC and the IP address. To verify certificates, each host requires the CA's public key. Typically it will be installed in the form of a self-signed certificate along with the host's own certificate, but a more complex Public Key Infrastructure (PKI) to support multiple LANs is also conceivable.

Basically, CLL divides all network traffic into four packet types: *ARP* and *DHCP*[1] control packets, *unicast* and *broadcast IP* data packets. Authentication is performed for all packet types and, in addition, an optional payload encryption is provided for unicast IP packets.

While ARP and broadcast IP packets are authenticated by means of public key cryptography (digital signatures in conjunction with certificates), unicast IP and DHCP packets get secured using fast symmetric key algorithms. Safeguarding unicast IP packets with a message authentication code and optionally a block cipher requires each pair of communicating hosts to share a secret key. For that purpose, CLL employs a key exchange protocol to negotiate shared keys on-demand. Since the IP traffic flow between two hosts always begins with an ARP exchange, CLL adopts it to establish a *security association (SA)* between the two peers. The two machines authenticate each other, negotiate a secret key and agree on the cryptographic algorithms to protect their IP packets. The establishment of an SA is subsequently referred to as *handshake*.

To determine the sender's (claimed) identity during the authentication of incoming packets, CLL examines the Ethernet header and, depending on the

[1] Though being encapsulated in an UDP segment and an IP datagram, we handle DHCP messages as a separate layer 3 packet type due to the functional position of DHCP below the network layer.

protocol, also the ARP, IP, or DHCP header. Where applicable, it performs a consistency check: the sender's MAC address can be also found in the ARP header or—in case of a DHCP client—in the DHCP header, and it must match the address specified in the Ethernet header. Such a cross-layer consistency check is not performed by other protocol layers. It is, however, crucially important to ward off ARP spoofing and DHCP starvation attacks. Layer 2 authentication alone would not suffice for this purpose.

The following listing summarizes the various LAN attacks fended off by CLL:

- *ARP spoofing:* impersonation, MiM and DoS attack
- *DHCP spoofing:* rogue DHCP server (MiM & DoS), DHCP starvation attack (DoS)
- *generic unicast attacks:* injection of spoofed packets, eavesdropping
- *generic broadcast attacks:* injection of spoofed packets, special case: smurf attack[2]

4 Cryptographic Design Decisions

The security philosophy of CLL is to provide the user with a suite of up-to-date cryptographic algorithms and corresponding parameters, letting her choose between them on her own responsibility. Such a design has the advantage of considering individual security perceptions, allowing to trade off between highest-level security and best performance, and supporting the prompt exchange of an algorithm being reported as broken. With our implementation, we nevertheless provide a reasonable default configuration to assist users without deeper understanding of cryptography. The general level of protection provided by CLL may be also selected. Either CLL just authenticates all types of packets or it additionally also encrypts the payload of unicast IP packets (including the IP header). Skipping the encryption step will result in a better performance and should be done whenever a higher layer security protocol like IPsec already ensures confidentiality. CLL allows to use different ciphers and hash functions in each direction of an SA. With regard to system complexity, we however prescribe the algorithms used for key exchange, key derivation, and DHCP packet authentication. Table 1 summarizes the algorithms proposed for CLL and supported by our implementation.

During the handshake CLL applies the Diffie-Hellman key agreement protocol to exchange a symmetric *master key* with perfect forward secrecy between the two peers. Since handshake packets are digitally signed, there exists no susceptibility to man-in-the-middle attacks. To the negotiated master key we apply a deterministic key derivation function to generate for each direction two properly sized keys—one for the message authentication code and one for the optional cipher.

[2] Flooding the victim via spoofed broadcast ping messages being answered by all other hosts.

Table 1. Algorithms and parameters in CLL

message auth. codes	• HMAC with MD5, SHA-160/256, RIPEMD-160 or HAS-160 • 128–256 bit key length
encryption	• optionally with a block cipher in CBC mode, 128–256 bit key length • available ciphers: Twofish, AES, RC6, RC5, Blowfish, MARS, Serpent, CAST-128/256, SEED, GOST
key exchange	Diffie-Hellman, 2048-bit group No. 14 from the IPsec specification
key derivation	IEEE 1363a Key Derivation Function 2 (KDF2) using RIPEMD-160
key rollover	periodically on demand, e. g., every 30 min
digital signatures	• RSA with variable key length (typically 1024–2048 bits) • RSASSA-PSS signature scheme with SHA-160/256 or RIPEMD-160
certificates	X.509 v3 with RSA signature, ASN.1 BER/DER encoding

CLL guarantees the authenticity of unicast IP and DHCP packets by means of a *Hashed Message Authentication Code (HMAC)* [4] attached to the end of each packet. In addition to authentication, CLL offers to protect unicast IP packets from eavesdropping by optionally encrypting them with a block cipher in *Cipher Block Chaining (CBC)* mode. When establishing an SA, we generate a random *Initialization Vector (IV)* and use afterwards the last encrypted block of the preceding packet as the next packet's IV. Since transmissions on the link layer are unreliable, the sender also prepends the current IV to each packet. If the payload size is not a multiple of the block size, random padding bytes are appended. We first encrypt the plaintext and then compute the HMAC for the ciphertext, since this is the only order that is generally considered secure [14]. It also enables to detect a forged packet without the need to decrypt it.

To sign handshake and broadcast IP packets, CLL applies the well-known RSA algorithm in conjunction with certificates. RSA offers the great advantage of supporting public key signatures and encryption with a single key pair. And though CLL's security architecture does not require any public key encryption, in practice the local CA can make use of RSA encryption to securely deploy the DHCP HMAC keys to the users.

5 Operation of CLL in Detail

5.1 Basic Packet Format

When securing Ethernet frames, CLL inserts its own headers and replaces the EtherType value in the Ethernet header with its own identifier (0x07D0, otherwise unassigned by IEEE). Figure 2 depicts the generic layout of an Ethernet frame safeguarded by CLL. The *CLL header* is placed behind the Ethernet header. It has been designed as a compact bit field to save overhead. It consists of a version number (currently 1) like in IP, a field specifying the encapsulated packet type (*unicast* or *broadcast IP packet*, *ARP handshake packet*, *DHCP client*

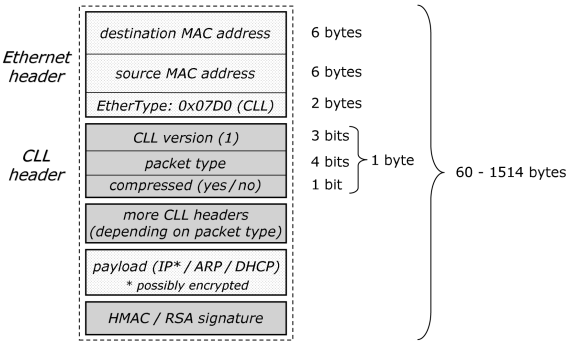


Fig. 2. An Ethernet frame in CLL

or *server packet*, internal *certificate packet*), and a Boolean flag stating whether the payload has been optionally compressed by CLL. This main CLL header is followed by one or more inner headers depending on the encapsulated packet's type. Therein we store, among cryptographic parameters, the original EtherType number. Behind the inner headers resides the payload, i. e., an ARP, IP, or DHCP packet. Finally, each secured Ethernet frame terminates with an authentication field containing either an HMAC (unicast IP and DHCP packets) or an RSA signature (ARP handshake and broadcast IP packets) computed over the whole frame.

5.2 ARP Handshake and SA Setup

Overview. To safeguard unicast IP packets, CLL needs to establish an SA between each pair of communicating hosts. For this, CLL takes advantage of the ARP mechanism and expands it at the same time with authentication. Figure 3 illustrates this ARP handshake between two hosts A and B.

When started, a CLL implementation should first flush the ARP cache, thus ensuring that all IP traffic to other hosts is preceded by an ARP request. Having intercepted the ARP request, CLL wraps it up into a digitally signed handshake

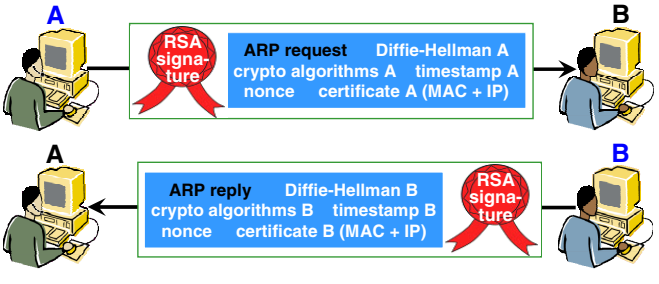


Fig. 3. ARP handshake: Diffie-Hellman key exchange in conjunction with RSA signatures

packet. It includes the host certificate and cryptographic parameters to establish the SA. The handshake packet is broadcasted like an ordinary ARP request and every station on the LAN checks whether it holds the inquired IP address. At the destination host, CLL verifies the certificate of the requesting host and validates the packet's signature. Invalid packets are dropped. Then it must be checked whether the sender's IP/MAC address pair claimed in the ARP request (and its MAC address stated in the Ethernet header) matches the one specified in its certificate.

If the handshake packet turns out to be valid, CLL creates a new SA with the requesting host, based on the local and the received cryptographic parameters. Finally, CLL strips off everything from the handshake packet except for the ARP header, restores the ARP EtherType number in the Ethernet header and passes the resulting ordinary ARP request up the protocol stack to the ARP module. The ARP module creates then an ARP table entry for the requesting host, and responds with an ARP reply. This reply gets intercepted again and is encapsulated into a digitally signed handshake packet analogously to the ARP request, along with the local cryptographic parameters and the host certificate. CLL then unicasts this second handshake packet to the requesting host like a usual ARP reply. In the following, we refer to the first handshake packet as the *handshake request* and to the second one as the *handshake reply*. On the requesting host the handshake reply undergoes the same verification process before the SA is established and the ARP reply is passed up to the ARP module.

Creating an SA implies the computation of a joint master key from the public and private Diffie-Hellman values. From the master key, CLL then derives the four keys for the HMAC and the optional block cipher. At any time, only one SA is permitted per host pair.

Handshake Packet Details. We employ a UNIX timestamp and a nonce to protect against replay attacks. CLL requires the clocks of all hosts on the LAN to be synchronized within reasonable limits decided on by the network administrator, e. g., in the range of 2–5 minutes. This can be easily achieved if the users manually adjust their computer's clock occasionally. An automatic clock synchronization, for instance by using NTP [17], is also possible after having established an SA to a trustworthy server.

The nonce is a random 64-bit number generated by the initiator of the handshake, which expects to find it repeated in the handshake reply. It ensures that the peer actually participates in the protocol, i. e., its handshake reply has not been replayed. Due to the nonce, it is not necessary to verify the timestamp in the handshake reply. It must, however, be stored for comparisons with timestamps of possibly future handshake requests.

The other important handshake element are the cryptographic parameters. Each host specifies the hash function configured for the HMAC and the block cipher potentially chosen to protect the payload against eavesdropping, along with the key sizes. A compression algorithm may be specified as well, if a host intends to compress its outgoing unicast IP packets. Moreover, each party states how long the SA should be valid before it is either extended or removed due

to inactivity. The actual SA validity period is the minimum of the two claims. However, it may not fall below a threshold currently set to 15 minutes to prevent permanent handshakes or renegotiations.

Retransmissions and Conflicts. CLL addresses the possibility of a handshake packet loss by means of retransmissions. In case of a lost (or just unanswered) handshake request the standard ARP retransmission mechanism will trigger a new ARP request. Having intercepted this ARP request, CLL retransmits the respective cached handshake request after updating its timestamp and signature. Through caching we avoid the computation-intensive generation of new Diffie-Hellman values.

The loss of a handshake reply will also result in a retransmission of the corresponding handshake request. The answering peer caches the received original handshake request as well as its own handshake reply. It is therefore able to recognize the incoming duplicate handshake request, and retransmits its handshake reply. Due to the receiver relying on the nonce, we can even omit to update timestamp and signature in this case.

Theoretically, it is conceivable that two hosts without an SA concurrently send each other a handshake request, when both of them have a pending IP datagram destined for the other one. However, only the creation of a single SA is allowed between two hosts. CLL resolves this issue by performing an integer comparison between the two 48-bit MAC addresses: the handshake request of the host with the higher MAC address "wins".

Complete and Incomplete SAs. From the point of view of a host, we refer to an SA as *complete* when it is known for sure that the peer has also established the SA. Host A as the initiator of an ARP handshake can set up the SA with its peer B only after having received the handshake reply. A's SA is therefore complete right from the start. Host A can immediately send secured unicast IP packets to its peer B and be certain that B will be able to verify and decrypt them.

In contrast, host B first has an *incomplete* SA, as long as it cannot be sure whether A has received its handshake reply. Usually, the IP datagram of host A that triggered the ARP request will quickly reach host B and thereby confirm the set up SA. However, as long as this is not the case, host B may not transmit any IP packets to its peer—A might not be able to authenticate them. Instead, in the unlikely case that B wants to transmit to A before A has sent the first packet, B must queue its IP datagram and send a new handshake request to A. This enforces the creation of a new SA, replacing the existing incomplete one.

Safeguarding against Replay Attacks. While the initiator of the SA protects itself against a replayed handshake reply with the aid of a nonce, its peer has to rely on the timestamp check when judging the freshness of an incoming handshake request. However, a timestamp is considered valid within a period of several minutes (smaller than the minimum SA duration) to tolerate time deviations. It hence does not assure a complete protection by itself. An attacker may try to replace an existing SA by replaying a captured outdated handshake

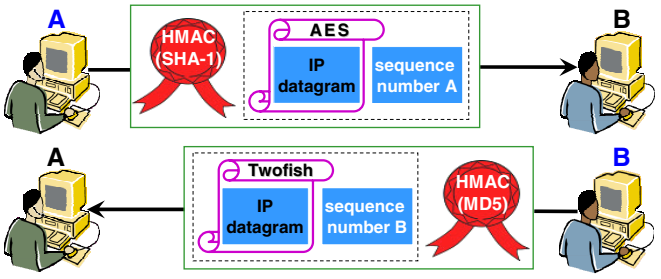


Fig. 4. Transmission of unicast IP packets safeguarded with a block cipher and a message authentication code

request bearing a timestamp which is still valid. CLL fends off such attacks by comparing the timestamp of a new handshake request with the timestamp of the handshake request or reply which led to the establishment of the currently existing SA. The use of timestamps avoids the necessity of a third message for a second nonce in the other direction, which would render the ARP handshake more complex.

5.3 Unicast IP Packets

Having created ARP table entries and established an SA, unicast IP packets can be transmitted between the two peers. This is illustrated in Figure 4. While host A encrypts its packets with the block cipher AES and authenticates them with an HMAC using the hash function SHA-1, its peer B employs Twofish and MD5. Taking the sender's MAC address the receiver looks up the corresponding SA to verify the packet's HMAC, sequence number, source IP address, and to decrypt the IP datagram. Only if the peer is a router, its IP address may differ from the source address stated in the IP header.

Each unicast IP packet contains a sequence number to protect against replay attacks. It is incremented by one with each packet sent to the respective destination. The receiver tolerates packet losses and only checks whether a packet's sequence number is larger than that of its predecessor. The sequence numbers are transmitted as plaintext to avoid an unnecessary decryption of replayed unicast IP packets. However, in order not to reveal the number of packets exchanged between two hosts so far, we generate the initial sequence numbers—one for each direction—at random.

Note that once having created an SA, CLL can also secure unicast packets carrying some other layer 3 protocol, e. g., Novell's IPX.

5.4 Periodical Key Rollover

By design, an SA has a short lifetime of typically 15–60 minutes like an ARP cache entry. But if any IP packets are transmitted during this period, it is renewed by a new Diffie-Hellman key exchange. New session keys for the HMAC

Fig. 5. Renegotiation—renewing an SA

and block cipher as well as sequence numbers are derived from a new master key. We call the extension of an SA *renegotiation*. Figure 5 illustrates the messages exchanged between two peers to extend their SA.

The *renegotiation request* and *reply* are the counterparts of the handshake request and reply. They are transferred through the existing SA like usual unicast packets. Each peer establishes a new SA after receiving the corresponding renegotiation packet. Just like when initially setting up an SA, host A's SA is complete from the beginning on, while host B first has an incomplete SA. But in case of a renegotiation, we cannot expect that an IP packet will be transmitted from A to B shortly and render B's SA complete as well. Therefore, host A has to explicitly acknowledge the reception of the renegotiation reply. It does so by means of a *renegotiation ack* sent through the new SA.

The renegotiation is initiated by the peer that first determines the expiration of the SA according to its clock. Concurrent renegotiation attempts are resolved in the same way as in the ARP handshake by performing a MAC address comparison.

During the renegotiation the peers re-exchange and re-validate their *current* certificates to address a possible expiration of the previous ones, especially in case of short-lived certificates issued via DHCP. While a renegotiation is in progress, pending IP packets destined for the peer can be still transferred through the old SA, i. e., there is no need to delay and queue them. We address the possibility of renegotiation packet losses by means of a retransmission mechanism.

5.5 Broadcast Packets

CLL authenticates broadcast IP packets like handshake packets by means of an RSA signature. To verify the signature, the receivers require the sender's host certificate. However, the variable payload size of a broadcast packet may well be too large to piggyback the certificate and still stay within the maximum segment size limit. Therefore, we broadcast the certificate in advance in a special *certificate packet*. CLL sends a certificate packet only when dispatching a broadcast packet and when more than a minute has passed since the previous certificate transmission, i. e., periodically on demand. All hosts on the LAN cache the received host certificates. Thus they need to validate each certificate only once and henceforth have the correct public key readily available for future signature verifications.

Like handshake packets, broadcast packets are protected against replay attacks by means of a timestamp combined with an additional counter. If a host sends more than one broadcast packet at the same UNIX time (i. e., within one

second), it increments this counter with each packet by one. All receivers store for each sender the timestamp and counter from its last broadcast packet. Subsequent packets from the same sender must bear a newer timestamp or the same timestamp with a higher counter value.

When dealing with high-rate broadcast traffic, the generation of RSA signatures on a per-packet basis may become computationally infeasible in real-time. However, in this case it is conceivable to queue outgoing broadcast packets for a short time and sign the accumulated group of packets as a whole with a single private key operation before dispatching them. The receivers would reassemble this group and verify the overall signature attached to the last packet. A sophisticated but also more complex approach tolerating packet losses might be the application of a specialized broadcast authentication protocol like TESLA [19].

6 Integrating and Securing DHCP

6.1 Basic Concept

So far, we have described the case of a static IP configuration, where the local CA creates for each machine a *host certificate* incorporating its MAC and IP address. However, CLL also supports the automatic assignment of IP addresses by means of DHCP. The DHCP message exchange is safeguarded and extended. CLL protects DHCP not only from unauthorized attackers, but also from malicious behavior originating from authenticated hosts.

In case DHCP is used, the local CA issues a *base certificate* for each host, bearing only the machine's MAC address and no IP address. The DHCP server uses the base certificate as a template to generate a full-fledged host certificate, which contains the assigned IP address. Thus, it acts as a second CA. The host certificate issued by the DHCP server has the same validity period as the IP address lease. When extending the DHCP lease, the host certificate is renewed accordingly.

Securing DHCP implies the authentication of all DHCP packets and a consistency check of the DHCP header in client-originated messages. Since CLL supervises the complete DHCP traffic in a transparent way, it also takes on the automatic application for a host certificate and its issuing. Its operation does not require any modifications on the employed DHCP client or server. On the client side, CLL attaches the base certificate to the DHCPREQUEST message. On the server CLL verifies this request and strips off its own headers, before passing it up to the DHCP module. It then waits for the outgoing DHCPACK message. This message constitutes the confirmation that the DHCP server has assigned the requested IP address. CLL extracts from it the allocated IP address along with the lease time, and issues a corresponding host certificate. Piggybacked on the DHCPACK message, the new host certificate finally reaches the client, which can now finalize its IP configuration and is ready to establish SAs.

6.2 Authenticating the Packets

We have designed the authentication of DHCP packets in a way that allows to employ an HMAC from the beginning, without requiring an initial public key

handshake. DHCP traffic occurs only between the clients and typically one single trusted server controlled by the local CA. Therefore, the number of communicating host pairs is limited and it is feasible to statically configure pre-shared HMAC keys. This task may be performed during the certificate enrollment without any additional effort. The local CA can generate a secret HMAC key for a host along with its base certificate. After encrypting the HMAC key with the host's public RSA key it can deliver these items to the user, e. g., via e-mail.

If the issued HMAC key were completely random, one would have to promptly configure it on the DHCP server as well, which involves some effort. Instead, we use a single *DHCP master key*, a concept adopted from [8]. From this master key we derive for each host the corresponding HMAC key by means of a key derivation function. The master key is known only to the local CA and the DHCP server. The pair <*MAC address*, *expiration time of the base certificate*>, in the following denoted as *client ID*, serves as the derivation parameter. This scheme does not require to inform the DHCP server about any newly certified hosts.

Since all hosts include their client ID into every sent DHCP packet, the DHCP server can deduce the corresponding HMAC key in an ad-hoc fashion and authenticate the packet. Conversely, when the DHCP server responds to the client, it has the right HMAC key already available. By incorporating the expiration time of the base certificate into the client ID we restrict the lifetime of the HMAC key. DHCP packets with expired client IDs are thus easily dropped without further verification. This allows, for instance, to immediately ignore DHCPDISCOVER messages sent by no longer authorized hosts.

To protect against replay attacks, we employ the same technique already introduced with broadcast packets, i. e., a UNIX timestamp in conjunction with a counter for packets bearing the same timestamp. A consistency check of the MAC and IP addresses stated in the DHCP header renders the authentication complete.

6.3 Further Security Measures

We consider the two DHCP messages DHCPDECLINE and DHCPRELEASE as a security risk. The first one allows a malicious client to spuriously tell the DHCP server that the IP address assigned to it is already in use by some other machine, thus making a DHCP starvation attack possible. The second one is utilized to release an assigned IP address to the DHCP server's address pool before the corresponding lease has expired. However, we cannot force a host to give up its certificate, and a malicious user might continue to use its certificate and with it the released IP address, while the address has also been assigned to some other machine. Therefore, we decided to simply drop these messages. Note that this does not violate the DHCP specification: these messages are transmitted in an unreliable manner without any retransmissions, i. e., they may get lost en route anyway. Moreover, no host is obliged to release its IP address ahead of time.

CLL allows to restrict the number of authenticated DHCP packets originated by the same host that the DHCP server will process during a specified time interval. Thereby the server can be secured against overload caused by malicious

or misconfigured clients, attempting to renew their IP address lease extremely often. This would force the server to continuously issue new host certificates, which includes an expensive private key operation.

These security measures prevent malicious behavior originating from authenticated hosts. Without them attacks on DHCP would be still feasible and one would have to extensively analyze the server's DHCP logfiles to backtrack the identity of the attacker.

7 Implementation and Evaluation

7.1 CLL as a Cross-Platform Service

We have implemented CLL in C++ as a *user-mode service* on both Windows (2000, XP, 2003, Vista) and Linux (kernels 2.4 and 2.6) using Visual C++ 2005 and GNU GCC 4.x respectively. Our CLL implementation consists of a platform independent core, which interoperates with a tailored portability layer providing a consistent interface for OS specific functionality. The responsibilities of the portability layer include crafting and filtering raw Ethernet frames, configuring the network interface (ARP, IP, MTU), and the interfaces for threads and timers.

To set up a filter handler for Ethernet frames in user-mode, we employ the packet filtering framework *WinpkFilter* [21] on Windows. On Linux, we have implemented a link layer filtering solution on our own. We unbind the real network adapter from the IP stack, transparently replace it with a virtual one (a tap device), and set up a raw PF_PACKET socket to send and receive Ethernet frames through the unbound real network adapter. A maybe somewhat more efficient kernel-level implementation of CLL's packet processing engine would constitute a complex and error-prone task, especially when targeting multiple platforms. We therefore leave it for future work. But despite the overhead of additional context switches, our user-mode approach achieves good performance, and is able to operate at wire-speed in 100 Mbit LANs. To support the large number of cryptographic algorithms proposed for CLL, we employ the comprehensive open source crypto library *Botan* [15].

Aiming to provide a real-world solution, we address in our implementation such issues like persistent storage of SA configurations (to tolerate an OS reboot) and backward compatibility. To support non-CLL capable devices like network printers or NAS and to enable a step-by-step migration, CLL can be configured to communicate with legacy hosts in the standard, insecure fashion. This is accomplished by providing the CLL-enabled hosts with a list of the legacy IP/MAC address pairs. CLL then sets up static ARP entries and thereby provides at least an unidirectional protection against ARP spoofing.

Since the drivers of common Wi-Fi adapters exhibit an Ethernet-compatible interface to the network stack, Wi-Fi networks can be secured by CLL as well.

7.2 Performance Evaluation

We have conducted a performance evaluation with two hosts A and B, where A is a laptop equipped with an AMD64 Turion 1.8 GHz CPU running Linux 2.6.20

Table 2. Performance of the ARP handshake

action	*duration in ms*
1st ping A → B using CLL: ARP handshake	27.4
1st ping A → B without CLL: usual ARP exchange	0.92
generating the private & public DH value (2048 bits)	host A: 26.3 host B: 44.1
deriving the master key with DH	host A: 7.2 host B: 15.7
computing an RSA-1024 signature	host A: 3.1 host B: 5.7

(32-bit) and B is a PC with an Intel Core 2 Duo E6400 2.13 GHz processor running Windows XP SP-2. The presented results are averaged over multiple runs.

The first series of measurements, shown in Table 2, is devoted to the overhead of the ARP handshake. For digital signatures both hosts use an RSA-1024 module. By pinging the neighboring host with no previously established SA we measure the time to perform the ARP handshake and the subsequent ICMP echo exchange. We compare it to the delay of the first ping in an ordinary unsecured setup, including a plain ARP message exchange.

Though it takes 30 times longer than a usual ARP exchange, the one-time delay of 27.4 ms induced by the ARP handshake with CLL is negligibly short for practical purposes. This low value is achieved due to an optimization in our implementation: we precompute the Diffie-Hellman values in a background thread, and thus have them readily available at the beginning of an ARP handshake. Otherwise the handshake would last $26.3 + 44.1 = 70.4$ ms longer. The delay of 27.4 ms can be broken down by measuring the computation time of the two dominating operations—the derivation of the master key with Diffie-Hellman and the creation of an RSA signature[3]. Deriving the master key is performed in parallel, thus taking $\max\{7.2, 15.7\} = 15.7$ ms, while signing is carried out sequentially and requires $3.1 + 5.7 = 8.8$ ms. Summing this up yields 24.5 ms. The remaining 2.9 ms are used for by the verification of the host certificates and handshake signatures, and also include the network round-trip time (RTT).

In the second series of measurements, we analyze the TCP throughput (using the tool *ttcp* [22]), the CPU load incurred at the sender and receiver, and the RTT between two hosts already sharing an SA. The results are shown in Table 3. When comparing the TCP throughput achievable with CLL to the result using a conventional, unsecured protocol stack, we observe only a very small decrease in speed of approximately 2 % without encryption and 3 % with encryption. It can be attributed quite exactly to the overhead induced by the additional CLL headers and fields. Encryption and authentication of packets with CLL apparently has virtually no effect on the achievable data rate in 100 Mbit LANs, which proves the feasibility of our approach.

[3] Though host B's CPU is faster than host A's CPU, the public key operations are slowed down by missing big integer assembler optimizations in Botan on Windows platforms.

Table 3. Performance of unicast transmissions in a 100 Mbit LAN

action	*measured values*			
TCP throughput using CLL:				
• *HMAC(MD5)*	A → B:	11 263 KB/s	55 / 26 % CPU (tx / rx)	
	B → A:	11 312 KB/s	22 / 60 % CPU (tx / rx)	
• *Twofish / HMAC(MD5)*	A → B:	11 113 KB/s	75 / 38 % CPU (tx / rx)	
	B → A:	11 160 KB/s	31 / 76 % CPU (tx / rx)	
TCP throughput without CLL	A → B:	11 522 KB/s	45 / 17 % CPU (tx / rx)	
	B → A:	11 519 KB/s	10 / 44 % CPU (tx / rx)	
RTT: 100 pings A → B using CLL	min: 0.287 ms	∅: 0.377 ms	max: 0.501 ms	σ: 0.046 ms
RTT: 100 pings A → B without CLL	min: 0.178 ms	∅: 0.198 ms	max: 0.231 ms	σ: 0.012 ms

By comparing the CPU utilization with and without CLL being used, we assess the induced additional CPU load. The overhead of piping the packets through the user-mode and computing the HMAC turns out to be entirely admissible. Even when enabling the block cipher, host A still has a quarter of its CPU time left for other tasks when processing packets at full wire-speed. The faster host B runs with a CPU utilization of only one third in the same situation. This machine obviously has the potential to operate CLL even in a Gigabit LAN, and to achieve a throughput of at least some hundred Mbit/s. Just like the TCP throughput, the RTT measured when running CLL in the Twofish / HMAC(MD5) configuration is very satisfactory. On average it is 0.38 ms, i. e., only twice the ordinary RTT without CLL. It should thus not represent a drawback for any typical application scenario.

8 Conclusion

In this paper, we have introduced the *Cryptographic Link Layer (CLL)*. CLL employs public key cryptography to identify all hosts in the Ethernet LAN based on their IP/MAC address pairs. It safeguards the packets transmitted between them against different spoofing attacks and eavesdropping. Pairs of hosts willing to communicate first establish security associations by an extension of the ARP handshake. In the course of this, the hosts authenticate each other, exchange cryptographic parameters, and negotiate symmetric session keys to protect their following unicast packets with a message authentication code and an optional cipher. Broadcast packets are also secured by CLL using digital signatures. When IP addresses are to be configured dynamically, CLL extends DHCP to automatically issue host certificates with the leased IP address. In the course of this, it also adds authentication to DHCP and safeguards it against various attacks.

We have implemented CLL on both Windows and Linux without modifying the existing protocol stack. Backward compatibility to ordinary, unsecured hosts can be enabled to support a step-by-step migration and retain legacy devices.

The evaluation of CLL demonstrated the excellent performance of our protocol in a 100 Mbit Ethernet LAN, where it achieved wire-speed throughput and short round-trip times.

References

[1] Altunbasak, H., Krasser, S., Owen, H., Sokol, J., Grimminger, J., Huth, H.-P.: Addressing the Weak Link Between Layer 2 and Layer 3 in the Internet Architecture. In: LCN 2004: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, November 2004, pp. 417–418 (2004)

[2] Antidote, `http://antidote.sourceforge.net`

[3] ArpWatch, `http://ee.lbl.gov` and `http://freequaos.host.sk/arpwatch`

[4] Bellare, M., Canetti, R., Krawczyk, H.: Message Authentication Using Hash Functions: the HMAC Construction. RSA CryptoBytes 2(1) (1996)

[5] Bruschi, D., Ornaghi, A., Rosti, E.: S-ARP: a Secure Address Resolution Protocol. In: ACSAC 2003: Proceedings of the 19th Annual Computer Security Applications Conference, December 2003, pp. 66–74 (2003)

[6] Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (April 2006)

[7] Droms, R.: Dynamic Host Configuration Protocol. RFC 2131 (March 1997)

[8] Droms, R., Arbaugh, W.: Authentication for DHCP Messages. RFC 3118 (June 2001)

[9] Ettercap, `http://ettercap.sourceforge.net`

[10] Gouda, M.G., Huang, C.-T.: A secure address resolution protocol. Computer Networks 41(1), 57–71 (2003)

[11] IEEE 802.1AE: Media Access Control (MAC) Security, `http://www.ieee802.org/1/pages/802.1ae.html`

[12] Jerschow, Y.I.: The CLL service & toolkit for Windows and Linux, `http://www.cn.uni-duesseldorf.de/projects/CLL`

[13] Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301 (December 2005)

[14] Krawczyk, H.: The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)

[15] Lloyd, J.: Botan Cryptographic Library, `http://botan.randombit.net`

[16] Lootah, W., Enck, W., McDaniel, P.: TARP: Ticket-based Address Resolution Protocol. Computer Networks 51(15), 4322–4337 (2007)

[17] Mills, D.L.: Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (March 1992)

[18] Montoro, M.: Cain & Abel, `http://www.oxid.it/cain.html`

[19] Perrig, A., Canetti, R., Tygar, J.D., Song, D.: The TESLA Broadcast Authentication Protocol. RSA CryptoBytes 5(2), 2–13 (2002)

[20] Plummer, D.C.: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. RFC 826 (November 1982)

[21] NT Kernel Resources: WinpkFilter, `http://www.ntkernel.com`

[22] Test TCP (TTCP) - Benchmarking Tool for Measuring TCP and UDP Performance, `http://www.pcausa.com/Utilities/pcattcp.htm`

[23] Vyncke, E., Paggen, C.: LAN Switch Security. Cisco Press (2007)

[24] Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (January 2006)

Faster Multi-exponentiation through Caching: Accelerating (EC)DSA Signature Verification

Bodo Möller[1,⋆] and Andy Rupp[2]

[1]Google
bmoeller@acm.org
[2]Horst Görtz Institute for IT Security
arupp@crypto.rub.de

Abstract. When verifying digital signatures, achieving a high throughput can be crucial. We present a technique that is useful for ECDSA and DSA signatures. It assumes that common domain parameters are used (which is typical of ECDSA) and that at least some signers recur (as in many application scenarios). We can achieve noticeable speedups in very different environments—from highly restricted ones where memory is very scarce to larger machines without severe memory restrictions. Requirements for the target platform are very small for a beneficial application of our technique. This makes it attractive for embedded systems, where ECDSA is a signature scheme of choice.

More generally, what we consider is the task of computing power products $\prod_{1\le i\le k} g_i^{e_i}$ ("multi-exponentiation") where base elements $g_2, \ldots, g_k$ are fixed while g_1 is variable between multi-exponentiations but may repeat, and where the exponents are bounded (e.g., in a finite group). We present a new technique that entails two different ways of computing such a product. The first way applies to the first occurrence of any g_1 where, besides obtaining the actual result, we create a cache entry based on g_1, *investing very little memory or time overhead.* The second way applies to any multi-exponentiation once such a cache entry exists for the g_1 in question and provides for a significant speed-up.

Keywords: Efficient implementation, elliptic curve cryptography, exponentiation, ECDSA, DSA, embedded cryptography.

1 Introduction

Consider a scenario where we repeatedly have to verify ECDSA signatures [1], trying to keep the computational delay small for each verification. A time-consuming step in ECDSA signature verification is computing a linear combination $u_1G + u_2Q$ of elliptic curve points G and Q, where G is specified by domain parameters (i.e., fixed) and where Q constitutes the signer's public key, with integers u_1 and u_2 in the interval $\big(0, \mathrm{ord}(G)-1\big)$ both depending on the specific signature. The same group with the same point G will typically be shared

⋆ Work done while the author was with the Horst Görtz Institute for IT Security.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 39–56, 2008.

by many signers since elliptic curve domain parameters are often taken from (intersecting) standards such as [19, Appendix 6], [1, Annex J], and [6] (with domain parameter specifications `NIST P-192` aka `prime192v1` aka `secp192r1` and `NIST P-256` aka `prime256v1` aka `secp256r1` common to all three of these). Also, we usually can expect some signers and thus their Q values to recur. For instance, consider public-key infrastructures:

- A verifying party will encounter end-entity certificates signed by possibly very many different intermediate certification authorities. When a new certification authority appears for the first time, the verifying party does not yet know how popular this particular certification authority is, i.e. if it has signed many or just very few end-entity certificates.
- The same applies to signatures on documents, such as the ECDSA signatures stored on the new digital "e-passports". When verifying a passport for airport immigration procedures, then quite possibly the next passenger in line may be using a passport signed by the same agency. On the other hand, the current passenger could be the only one from this particular country for days.

Thus, for a given G, we have to compute linear combinations $u_1G + u_2Q$ where Q sometimes is "new" and sometimes is "old" (has been seen before); but when a new Q appears, we generally do not know if and how frequently it will reappear.

There are well-known techniques to compute $u_1G + u_2Q$ much faster than by computing both u_1G and u_2Q individually, and this can be done yet faster if G and Q are both fixed and a one-time precomputation depending on these points has been done. Performing such precomputation whenever a "new" Q shows up may pay out if Q turns out to repeat, so that G and Q are fixed for a number of linear combinations. However, this is an investment of resources that would be lost if this particular Q does in fact not repeat.

We present a new technique that nearly avoids this drawback, provided that space for *permanently fixed* precomputation depending on G only is not severely limited. The first occurrence of some point Q in a computation $u_1G + u_2Q$ will incur very little penalty in terms of memory or time, and yet will leave behind useful precomputed data that can be cached to speed up subsequent linear combination computations involving the same G and Q.

While the ECDSA scenario best illustrates the practical use of our technique[1], the approach is in fact not restricted to the ECDSA or DSA case but may be

[1] Another approach to speed up ECDSA signature verification is due to Antipa et al. [2,24]. It works best for a slightly modified variant of the original signature scheme, dubbed ECDSA*, but under appropriate circumstances, it can be useful for the verification of standard ECDSA signatures. Where it makes sense to use the technique from [2,24], our technique may be preferable depending on the expected proportion of "old" to "new" Q values. In fact, we can get some of the benefit of [2,24] for any "new" Q and all of the benefit of our technique for any "old" Q by using a combination of both techniques in the case of a "new" Q, using specific imbalanced-scalar parameterizations within [2,24]. We omit further details on this.

applied in more general settings: It is suitable for any abelian group or (more generally) abelian semigroup with an identity element, henceforth written multiplicatively so that what we just described as linear combinations now turns into power products. Computing power products sometimes is called *multi-exponentiation* since it is a generalization of computing powers (exponentiation). The computational task that we will consider is computing power products of the form

$$\prod_{1 \le i \le k} g_i^{e_i}$$

where base elements $g_2, \ldots, g_k$ are fixed once and for all, whereas g_1 is variable between multi-exponentiations and *may* repeat, while all exponents e_i are assumed to be ephemeral. We will assume that the exponents are positive and at most ℓ bits long. (An appropriate value of ℓ is usually implied by the group order. A negative exponent for a group can be handled through inversion of the base element, or by reduction of the exponent modulo o where o is some multiple of the order of the base element, such as the group order.) For our technique to work as intended, we also assume that the exponent to variable base g_1 is not pathologically short (i.e., its length not just a fraction of ℓ); rare statistical outliers are no problem. Besides ECDSA signature verification, this setting also covers DSA signature verification [19]; however, it only applies when using common domain parameters, which is much more customary for ECDSA.

Concerning the implementation platform, we only need to make very mild assumptions that are, in particular, commonly satisfied by *embedded systems:* We assume that at least read-only memory is not severely limited, so that precomputation depending on $g_2, \ldots, g_k$ can be permanently stored. We also assume that some memory is available for caching at least one group element with an integer. Such data will be put into cache memory when performing a multi-exponentiation $\prod_{1 \le i \le k} g_i^{e_i}$ involving a "new" g_1 (i.e., one for which no cache entry currently exists), and can be used to speed up any subsequent multi-exponentiation repeating the same g_1 as long as the cache entry is kept. While the method is easier to describe assuming that dedicated cache memory is available, Appendix C will show that the technique can be quite useful even if this is not the case and a portion of fast read/write memory has to be sacrificed instead: In a specific example scenario where read/write memory is very scarce (which is typical of smart cards and other embedded devices), our technique already leads to a 10 % average speed advantage. The technique is also useful for devices without such constraints; the specific speed advantage factor gained by using our method strongly depends on the concrete application scenario and can be significantly higher than the 10 % in said example.

Like many approaches for exponentiation using precomputation (such as the Lim/Lee approach [12]), our technique has roots that can be traced back to Pippenger [21,22]; see also [4]. The novelty here in this regard is that for cached precomputation, we do not store powers of the form $g_1^{2^n}$ as by [21], which would

impose some computational overhead while computing $\prod_{1\le i\le k} g_i^{e_i}$ when g_1 is new. Instead, we store other powers of g_1 that happen to come up without effort if we arrange the computation suitably.

Section 2 describes preliminaries for our technique: interleaved multi-exponentiation, and radix-2 exponent splitting. Then, Section 3 presents the novel multi-exponentiation technique, which relies on caching certain intermediate results that can be obtained by investing very little additional read/write memory or time, allowing us to speed up later multi-exponentiations if an appropriate cache entry is available. Section 4 gives example performance figures for the new technique in certain application scenarios. Appendix A provides a comprehensive example to illustrate the technique. Appendix B discusses some implementation aspects. Finally, Appendix C considers a particular scenario to demonstrate the performance gain achievable by using the new technique.

2 Multi-exponentiation

We show known techniques that we later will use and combine in a novel way. Section 2.1 describes *interleaved multi-exponentiation,* an approach for computing power products. It also briefly describes some properties of radix-2 exponent representations that can be used in interleaved multi-exponentiation. Section 2.2 describes the technique of *radix*-2 *exponent splitting,* which can be used to obtain shorter exponents by converting exponentiation tasks into multi-exponentiation tasks, or converting k-fold multi-exponentiation tasks into k'-fold multi-exponentiation tasks with $k' > k$. Radix-2 exponent splitting is a useful technique for fixed bases (namely, for exponentiation or multi-exponentiation with precomputation that can be done in advance).

2.1 Interleaved Multi-exponentiation

We build on the straightforward multi-exponentiation strategy that has been called *interleaving* in [14], which generalizes well-known methods for single exponentiations such as the (left-to-right) binary or sliding window methods. Assume that radix-2 representations $e_i = \sum_{0\le j\le \ell} b_{i,j}\cdot 2^j$, $b_{i,j} \in B_i$, of all exponents are given where each B_i is a set of integers. We write $e_i = (b_{i,\ell}, b_{i,\ell-1}, \ldots, b_{i,1}, b_{i,0})_2$ and call the $b_{i,j}$ *digits* and B_i a *digit set.* We require that every g_i^b for $b \in B_i \setminus \{0\}$ be available from precomputed data. Note that in a group where inversion is particularly easy (such as those used in elliptic curve cryptography where an inversion requires just obtaining an additive inverse in the underlying field or performing a field addition), obtaining g_i^{-b} from precomputed data is easy if g_i^b has been stored; so both b and $-b$ can be included in set B_i if the single element g_i^b has been stored in a precomputed table of powers. In this setting, interleaved multi-exponentiation computes the power product as follows.

$A \leftarrow 1_G$ {Start with identity element}
for $j = \ell$ down to 0 **do**
  $A \leftarrow A^2$
  for $i = 1$ to k **do**
    if $b_{i,j} \neq 0$ **then**
      $A \leftarrow A \cdot g_i^{b_{i,j}}$ {Multiply by [inverse of] precomputed element}
return A

This is a left-to-right technique in that it proceeds from the most significant digits ("left") down to the least significant digits ("right").

Typical digits sets are of the form $B^{\pm}(m) = \{\pm 1, \pm 3, \pm 5, \pm 7, \ldots, \pm m, 0\}$ for groups where inversion is easy, or $B(m) = \{1, 3, 5, 7, \ldots, m, 0\}$ for semigroups in general. Here parameter m is an odd integer, often but not necessarily of the form $(11\ldots11)_2$, i.e. $m = 2^w - 1$, $w \geq 1$ an integer. This special form applies to the sliding window technique (cf. [9]) and to various variants of it that employ signed-digit representations of exponents, such as those introduced in [13] using a right-to-left conversion from binary to signed-digit representation and in [17,3,20] using left-to-right conversions. The general case with an arbitrary odd m was introduced as *fractional window representations* in [15], with left-to-right conversions for the signed-digit case suggested in [11,23,16]. Different digits sets can be used for different exponents, so we have $B_i = B(m_i)$ or $B_i = B^{\pm}(m_i)$ with per-exponent parameters m_i when employing such representations.

The *length* of a representation is the number of digits that remain if we drop any leading zeros (so the length of $(b_\ell, b_{\ell-1}, \ldots, b_1, b_0)_2$ is $\ell + 1$ if $b_\ell \neq 0$). Maximum length $l + 1$ is sufficient to represent any l-bit integer e ($2^{l-1} \leq e < 2^l$) in any of the representations mentioned above [18] (length l is sufficient for any of the unsigned-digit representations), and the minimum length with these representations is $l + 1 - \lceil \log_2 m \rceil$. Thus, the maximum outer loop index ℓ in the algorithm as shown above is sufficient for integers up to ℓ bits.

The *weight* of a representation is the number of digits that are non-zero. The conversion techniques mentioned above are known to achieve, for any integer e, the minimum weight possible given the respective digit set [18,16]. For unsigned and signed fractional window representations using digit set $\{1, 3, 5, \ldots, m, 0\}$ or $\{\pm 1, \pm 3, \pm 5, \ldots, \pm m, 0\}$, the average weight for random integers up to ℓ bits is slightly above

$$\frac{\ell}{w(m) + \frac{m+1}{2^{w(m)}}} \quad \text{and} \quad \frac{\ell}{1 + w(m) + \frac{m+1}{2^{w(m)}}},$$

respectively, where $w(m) = \lfloor \log_2(m+1) \rfloor$; for the average *density* (weight divided by ℓ), we have convergence to the resulting estimates as ℓ goes to ∞ (see [16]). For the special case $m = 2^w - 1$ (i.e., the sliding window technique and its non-fractional signed-digit counterparts), such that $w(m) = w$, the above is simply $\ell/(1 + w)$ and $\ell/(2 + w)$, respectively.

Observe that in the interleaved multi-exponentiation algorithm as shown above, (semi-)group operations need not actually be performed until after the

first multiplication of A by a precomputed element or its inverse, since $A = 1_G$ holds up to this point. This means that the initial squarings of 1_G can be skipped, and the first operation $A \leftarrow A \cdot g_i^{b_{i,j}}$ amounts to an assignment $A \leftarrow g_i^{b_{i,j}}$.

To estimate the time required to perform an interleaved multi-exponentiation, we thus need the maximum length of the representations of the e_i to determine the number of squarings, and the weight of the representation of each e_i to determine the number of other multiplications by elements available from precomputed data. (The maximum length is one more than the number of squarings, and the sum of the weights is one more than the number of other multiplications.) This is not counting any group inversions, since we would only use these in the algorithm if inversion is easy. In addition to this, we have to look at the time needed for precomputation. If g_i is a fixed base, we can assume that the required powers g_i^b have been precomputed in advance (and possibly built into ROM) and thus do not enter the time estimate. However, if g_i is not fixed, some effort goes into precomputing these powers: from g_i, the powers $g_i^3, \ldots, g_i^{m_i}$ can be computed using one squaring (to obtain g_i^2 as an intermediate value) and $\frac{m_i-1}{2}$ other multiplications.

(Note that the minimum-weight property of a conversion technique does not mean that it always provides the best radix-2 representation possible given the particular digit set. As discussed in [15, Section 5.1] and [16, Section 4], *modified signed fractional window representations* can sometimes reduce length without increasing weight. In certain situations, it may even be of advantage to accept a slight increase of weight for the sake of length reduction if saved squarings [due to length reduction] outweigh the additional multiplications [due to weight increase]. To pursue this approach, we can generalize the concept of radix-2 representations: e.g., $(100000)_2 = 2^5$ could be converted into $3 \cdot 2^2 + 5 \cdot 2^2$, which is not a proper radix-2 representation but might be written as $\binom{3}{5}00\Big)_2$ using a "double digit" of weight 2. Details are out of the scope of the present paper; we just mention this as a reminder that minimum-weight radix-2 representations can sometimes be improved by applying appropriate substitution rules.)

2.2 Radix-2 Exponent Splitting

We have seen that the length of exponent representations is important to efficiency since it determines the number of squarings needed for interleaved multi-exponentiation. For an exponent e, this length is around $\log_2 e$ with any of the representations mentioned in Section 2.1 as long as parameter m is reasonably small. *Radix-*2 *exponent splitting,* shown in the following, is a simple but effective idea (underlying [5] and made explicit in [8]) to get better performance if all bases are fixed.

For exponentiations g^e with exponent representations $e = (b_\ell, \ldots, b_0)_2$ of maximum length $\ell + 1$, we can decide to split each such representation into some number s of shorter exponent representations. To wit, let $\ell + 1 = L_1 + \cdots + L_s$ with positive integers $L_i \approx \frac{\ell+1}{s}$, and then let $e_1 = (b_{L_1-1}, \ldots, b_0)_2$, $e_2 = (b_{L_1+L_2-1}, \ldots, b_{L_1})_2$, and so on:

$$e = (b_\ell, \ldots, b_0)_2 = (\underbrace{b_{L_1+\cdots+L_s-1}, \ldots, b_{L_1+\cdots+L_{s-1}}}_{e_s},$$
$$\underbrace{b_{L_1+\cdots+L_{s-1}-1}, \ldots, b_{L_1+\cdots+L_{s-2}}}_{e_{s-1}}, \quad \ldots, \quad \underbrace{b_{L_1-1}, \ldots, b_0}_{e_1})_2$$

Then from $e = \sum_{1 \le i \le s} e_i \cdot 2^{L_1+\cdots+L_{i-1}}$ it follows that

$$g^e = g^{e_1} \cdot \left(g^{2^{L_1}}\right)^{e_2} \cdot \ \cdots \ \cdot \left(g^{2^{L_1+\cdots+L_{s-2}}}\right)^{e_{s-1}} \cdot \left(g^{2^{L_1+\cdots+L_{s-1}}}\right)^{e_s},$$

and thus by defining $g_i = g^{2^{\sum_{1 \le I < i} L_I}}$ we have transformed the task of computing g^e into the s-fold multi-exponentiation $\prod_{1 \le i \le s} g_i^{e_i}$. There is no need to actually evaluate the e_i as integers here since we already have appropriate representations of them—namely, portions of the original representation as shown.

Thanks to exponent splitting, the maximum length of exponent representations can go down from $\ell + 1$ to $\lceil \frac{\ell+1}{s} \rceil$ if the L_i are chosen accordingly. If g is fixed (and the parameters L_i are constant), then so are the g_i as defined here. Thus, the powers g_i^b needed by the interleaved multi-exponentiation algorithm in Section 2.1 can be precomputed in advance. So using additional memory for precomputed data (possibly ROM) allows us to save time in each exponentiation.

So far, we have looked at radix-2 exponent splitting applied to exponentiation, not to multi-exponentiation: each single exponentiation is converted into a multi-exponentiation. Radix-2 exponent splitting can just as well be applied for any fixed base in multi-exponentiation tasks, converting a k-fold multi-exponentiation into some k'-fold multi-exponentiation, $k' > k$. However, since the exponent splitting technique needs additional precomputed data (the powers $g_i = g^{2^{\sum_{1 \le I < i} L_I}}$ of base g), it cannot be used to advantage for bases that are not fixed. Thus, if there is any base that is not fixed (as in the case of DSA and ECDSA signature verification), long exponent representations may remain, and radix-2 exponent splitting hence will typically provide essentially no speed advantage in this situation.

3 Faster Multi-exponentiation by Caching Intermediate Results

This section describes a novel technique for computing power products $\prod_{1 \le i \le k} g_i^{e_i}$ assuming that $g_2, \ldots, g_k$ are fixed base elements, while g_1 is a variable base element whose values may recur. The technique is based on interleaved multi-exponentiation and on exponent splitting, but adds new features. It consists of two different multi-exponentiation algorithms. The first algorithm, described below in Section 3.1, is employed whenever a "new" g_1 value appears. This algorithm not only computes the multi-exponentiation result, but also outputs certain intermediate results, intended to be cached for later use. The second algorithm, described below in Section 3.2, can be employed whenever an "old" g_1

value appears, namely one for which a cache entry already exists. This algorithm then exploits the cache entry created by the first algorithm to compute the new multi-exponentiation result faster.

For both algorithms, we assume that parameters for radix-2 exponent splitting have been fixed, i.e. we have constant integers s and $L_1, \ldots, L_s$ as described in Section 2.2, used identically for all bases $g_2, \ldots, g_k$. We demand that $L_1 + 1 \geq \max_{1 \leq i \leq s} L_i$. For these bases, we furthermore assume that digit sets for exponent representations have been fixed (see Section 2.1), and that there is a fixed length limit $\ell{+}1$ for exponent representations. (This is enough for exponents up to ℓ bits, using any of the representations mentioned in Section 2.1.) We also require that powers of $g_2, \ldots, g_k$ as required for radix-2 exponent splitting using the given digit sets and exponent splitting parameters are precomputed in advance. These are constant elements, so they may be stored in ROM. Due to our assumption that at least read-only memory is not severely limited, it should be possible to store quite a number of such precomputed elements, allowing us to use reasonably large digit sets in the representations of exponents $e_2, \ldots, e_k$ that will undergo radix-2 exponent splitting.

Of course, since cache entries take up read/write memory, they eventually may have to be expired as new g_1 values occur. Once the cache entry for a certain g_1 has been deleted, this particular value again will have to be considered "new" if it occurs once more later. In extreme cases, the cache might provide space just for a single cache entry. Then, depending on the caching strategy implemented, g_1 might be recognized as "old" only if two immediately consecutive multi-exponentiations involve the same g_1 value, since any new value might lead to an instant cache eviction to make space for a new entry. However, it would also be possible to keep the existing cache entry for a while even if new g_1 values appear, meaning that any cacheable data created for such a new g_1 value would have to be discarded for lack of space. Which caching strategy is to be preferred depends very much on the statistical properties of the application scenario.

3.1 Multi-exponentiation for a New Base g_1

If no cache entry based on g_1 is available, $\prod_{1 \leq i \leq k} g_i^{e_i}$ should be computed as follows. As in Section 2.1, we assume that the exponents are given in representations $e_i = \sum_{0 \leq j \leq \ell} b_{i,j} \cdot 2^j$, $b_{i,j} \in B_i$.

First, apply radix-2 exponent splitting (Section 2.2) to the representations of exponents e_2 through e_k such that all of the resulting exponent representations observe maximum length $L = \max_{1 \leq i \leq s} L_i$ ($\approx \frac{\ell+1}{s}$). This transforms the k-fold multi-exponentiation task into a multi-exponentiation task with more bases, where the exponent to g_1 appears unchanged but all other exponent representations have been split into parts no longer than L digits. The list of bases has expanded from $(g_1, g_2, \ldots, g_k)$ into

$$\big(g_1, \quad g_2, g_2^{2^{L_1}}, \ldots, g_2^{2^{L_1+\cdots+L_{s-1}}}, \quad \ldots, \quad g_k, g_k^{2^{L_1}}, \ldots, g_k^{2^{L_1+\cdots+L_{s-1}}}\big);$$

we will assume that g_1 keeps its index ($i = 1$). Now apply the interleaved multi-exponentiation algorithm from Section 2.1 to this new $\big(1 + (k-1)s\big)$-fold power

product. (Remember that appropriate precomputed powers of the bases except g_1 are assumed to be available e.g. from ROM.) This will generate the desired result, $\prod_{1\le i\le k} g_i^{e_i}$. Additionally, it will generate certain intermediate values that turn out to be very useful.

Observe that no loop iteration before $j = L_1$ may involve non-zero exponent digits for any base other than g_1 (since we have $L_1 + 1 \ge L_i$ for any of the exponent splitting parameters $L_2, \ldots, L_s$). In other words, before this round, A has never been multiplied with a power of a base other than g_1 . In particular, we have $A = g_1^{(b_{1,\ell},\ldots,b_{1,L_1})_2}$ just after the inner loop iteration for $j = L_1, i = 1$ (and still after the outer loop iteration for $j = L_1$ if $L_1 \ge \max_i L_i$). From this and earlier loop iterations, we can obtain the following $s-1$ intermediate values:

$$\begin{aligned} j = L_1 + \cdots + L_{s-1} \;&\Rightarrow\; A = g_1^{(b_{1,\ell},\ldots,b_{1,L_1+\cdots+L_{s-1}})_2} \\ &\cdots \qquad \cdots \\ j = L_1, i = 1 \;&\Rightarrow\; A = g_1^{(b_{1,\ell},\ldots,b_{1,L_1})_2} \end{aligned}$$

Thus, we can output the following data to be cached—a *cache entry* comprising g_1 itself (as index to the cache) and $s-1$ pairs, each consisting of an integer and the corresponding power of g_1:

$$\begin{gathered} \Big(g_1,\ \Big((b_{1,\ell},\ldots,b_{1,L_1})_2,\ g_1^{(b_{1,\ell},\ldots,b_{1,L_1})_2}\Big), \\ \ldots, \\ \Big((b_{1,\ell},\ldots,b_{1,L_1+\cdots+L_{s-1}})_2,\ g_1^{(b_{1,\ell},\ldots,b_{1,L_1+\cdots+L_{s-1}})_2}\Big)\Big) \end{gathered}$$

Note that when writing this to cache, the integers may be evaluated as such—there is no need to store the specific radix-2 representations. (However, since all of these integers are derived from e_1 following a fixed rule, it is clear that at most ℓ bits are sufficient to store complete information on all of them, should memory efficiency be an utmost concern.) With any of the representations mentioned in Section 2.1, these partial integers are guaranteed to be non-negative, with

$$(b_{1,\ell},\ldots,b_{1,L_1})_2 \;\ge\; \ldots \;\ge\; (b_{1,\ell},\ldots,b_{1,L_1+\cdots+L_{s-1}})_2 \;\ge\; 0.$$

Furthermore, if e_1 is uniformly random from some set $(0,\ldots,q)$ of integers where q is an ℓ-bit integer, then (unless L_s is very small) all of these integers will actually be positive with high probability (and will be reasonably close to $2^{\ell-L_1},\ldots,2^{\ell-L_1-\cdots-L_{s-1}}$, respectively; i.e., since $\ell = \sum_{1\le i\le s} L_i$, to $2^{L_2+\cdots+L_s},\ldots,2^{L_s}$).

Depending on the assumed distribution of e_1, it may be a good idea to skip writing a cache entry if it ever turns out that $(b_{1,\ell},\ldots,b_{1,L_1+\cdots+L_{s-1}})_2 = 0$. In any case, writing a cache entry should be skipped if all of the integers in it would be zero (and thus the corresponding powers of g_1 trivial).

3.2 Multi-exponentiation for an Old Base g_1

If a cache entry based on g_1 is available (created as described in Section 3.1), then $\prod_{1\le i\le k} g_i^{e_i}$ may be computed as follows. First, parse the cache entry as $\big(g_1,\ (\lambda_1, G_1),\ \ldots,\ (\lambda_{s-1}, G_{s-1})\big)$. Here we have $G_i = g_1^{\lambda_i}$ for $1 \le i \le s-1$, and if one of the exponent representations mentioned in Section 2.1 was used while creating the cache entry as specified in Section 3.1, we have $\lambda_1 \ge \ldots \ge \lambda_{s-1} \ge 0$. Now split e_1 into integers E_i $(1 \le i \le s)$:

- let $d_0 = e_1$;
- for $1 \le i \le s-1$, let $E_i = \lfloor \frac{d_{i-1}}{\lambda_i} \rfloor$ and $d_i = d_{i-1} - E_i\lambda_i$;
- and finally, let $E_s = d_{s-1}$.

In the exceptional case that $\lambda_i = 0$, $E_i = 0$ should be substituted for $\lfloor \frac{d_{i-1}}{\lambda_i} \rfloor$. By this construction, we have $e_1 = E_1\lambda_1 + \cdots + E_{s-1}\lambda_{s-1} + E_s$. It follows that

$$g_1^{e_1} = G_1^{E_1} \cdot \ \cdots\ \cdot G_{s-1}^{E_{s-1}} \cdot g_1^{E_s},$$

and thus we have transformed the power $g_1^{e_1}$ into a power product using new exponents E_i. This step is similar to radix-2 exponent splitting; we call it *modular exponent splitting.* Suitable digit sets for radix-2 representations of each of the new exponents can be chosen depending on how much read/write memory is available for storing powers of the bases $G_1, \ldots, G_{s-1}$ and g_1 (cf. Section 2.1).

For the exponents to the fixed bases $g_2, \ldots, g_k$, we again (exactly as in Section 3.1) assume that these are given in representations $e_i = \sum_{0\le j\le \ell} b_{i,j} \cdot 2^j$, $b_{i,j} \in B_i$. We apply radix-2 exponent splitting to these, giving us exponent representations of maximum length L.

In total, by applying both modular exponent splitting and radix-2 exponent splitting, we have converted the k-fold multi-exponentiation into a ks-fold multi-exponentiation. The maximum length of exponent representations here may exceed L since we do not have strict guarantees regarding the E_i. However, under the assumptions regarding the distribution of e_1 stated in Section 3.1, the maximum length will remain around L with high probability.

This completes the description of our new technique. For an illustrative example we refer the reader to Appendix A.

4 Performance

Our multi-exponentiation technique can be used under many different parameterizations—the number of bases may vary; the length of exponents may vary; the amount of memory available for fixed precomputation (such as ROM) may vary; the amount of memory available for cache entries (such as slow read/write memory) may vary; the amount of memory available for variable precomputed elements (i.e., intermediate powers) needed by the interleaved exponentiation algorithm may vary; and under any of these parameterizations, we have to decide on parameters s and $L_1, \ldots, L_s$ for exponent splitting (s-fold

exponent splitting with exponent segment lengths L_i), and we have to decide on digit sets and representation conversion techniques for the exponents to the fixed bases $g_2, \ldots, g_k$ on the one hand, and for *any* of the s partial exponents created from e_1 when the algorithm from Section 3.2 uses a cache entry on the other hand. This encompasses a large variety of different settings.

In the present section, we will look at a specific range of rather simple use scenarios for our new technique to assess its performance. Let us assume that we want to implement the multi-exponentiation technique in an environment where only a very limited amount of fast read/write memory is available but where we have some slower memory suitable for the cache, and where we have plenty of read-only memory for permanently fixed precomputed elements. As powers of g_1 are frequently needed in the course of the algorithm, this is what we will use such fast memory for. As particular examples, let us consider the cases where we have such fast memory space to store 4, 8, 16 or 32 group elements, and let ℓ be 160, 192 or 256, which are practical values for ECDSA. Note that restricting the space for storing powers of a base also limits the number of different digit values that we can use in exponent representations for the interleaved multi-exponentiation algorithm. We have implemented our new multi-exponentiation strategy and counted certain group operations under these prerequisites for different values of the splitting parameter s, always using reasonable $L_i \approx \frac{\ell+1}{s}$ and a left-to-right signed fractional window representation using appropriate digit sets $B^{\pm}(m) = \{\pm 1, \pm 3, \ldots, \pm m, 0\}$ such as to fully utilize the fast memory. (See [11,23,16] for details regarding left-to-right signed fractional window conversions.)

We have repeatedly simulated the behavior of our technique for uniformly random exponents in the interval $(0, \ldots, 2^{\ell} - 1)$, covering both the case of "new bases" to create cache entries (Section 3.1) and the case of "old bases" to observe the performance given such cache entries (Section 3.2). In these simulations, we have counted the following operations:

- Squarings (S) and other multiplications (M) used for precomputing powers of g_1 (including powers of cache entries derived from g_1);
- squarings (S) and multiplications (M) by precomputed powers of g_1 (or of cache entries) within the interleaved multi-exponentiation algorithm.

We have excluded from counting any of the multiplications by fixed precomputed elements (from ROM), since these are not a limiting factor given the assumption that plenty of space is available for these elements: low-weight exponent representations accordingly may be used for the corresponding exponents, so changes of the parameterization have less of an impact here. (Refer to Section 2.1 for applicable weight estimates.) The simulation results can be found in Table 1. The values in the first row ($s = 1$) reflect the special situation when no splitting at all is done. This applies to the multi-exponentiation algorithm for a new base g_1 for which no cache entry is available (Section 3.1), where a signed-digit representation is used for the full-length exponent e_1. The remaining rows contain operation counts for cases where g_1 is an old base, i.e., an existing cache entry is used (Section 3.2). As we can see from the table, the number of squarings will be reduced to about

Table 1. Experimental performance figures (squarings and multiplications with powers of g_1) for s-fold exponent splitting with exponents up to ℓ-bits, with space for 4, 8, 16, or 32 elements for variable precomputation

		$\#var = 4$	$\#var = 8$	$\#var = 16$	$\#var = 32$
$s = 1$	precomp.	$1S + 3M$	$1S + 7M$	$1S + 15M$	$1S + 31M$
	$\ell = 160$	$159.9S + 31.5M$	$156.0S + 26.1M$	$155.0S + 22.3M$	$154.0S + 19.5M$
	$\ell = 192$	$188.9S + 37.9M$	$187.9S + 31.4M$	$187.0S + 26.9M$	$186.0S + 23.5M$
	$\ell = 256$	$252.9S + 50.6M$	$251.9S + 42.1M$	$251.0S + 36.0M$	$250.0S + 31.5M$
$s = 2$	precomp.	$2S + 2M$	$2S + 6M$	$2S + 14M$	$2S + 30M$
	$\ell = 160$	$79.5S + 39.9M$	$79.1S + 32.0M$	$78.6S + 26.8M$	$78.6S + 23.2M$
	$\ell = 192$	$95.7S + 47.9M$	$94.7S + 38.5M$	$95.0S + 32.2M$	$93.7S + 27.8M$
	$\ell = 256$	$127.6S + 63.9M$	$126.9S + 51.4M$	$126.6S + 42.8M$	$126.8S + 36.8M$
$s = 3$	precomp.	$1S + 1M$	$3S + 5M$	$3S + 13M$	$3S + 29M$
	$\ell = 160$	$54.7S + 49.4M$	$53.4S + 37.4M$	$52.8S + 30.5M$	$52.4S + 25.9M$
	$\ell = 192$	$64.3S + 59.1M$	$63.3S + 44.7M$	$62.9S + 36.6M$	$62.3S + 31.1M$
	$\ell = 256$	$85.8S + 78.6M$	$85.0S + 59.6M$	$84.5S + 48.5M$	$84.5S + 41.1M$
$s = 4$	precomp.	$0S + 0M$	$4S + 4M$	$4S + 12M$	$4S + 28M$
	$\ell = 160$	$40.6S + 53.9M$	$40.7S + 40.7M$	$39.3S + 32.8M$	$38.8S + 27.8M$
	$\ell = 192$	$48.4S + 64.5M$	$47.6S + 48.6M$	$47.6S + 39.2M$	$46.9S + 33.1M$
	$\ell = 256$	$64.1S + 85.7M$	$64.1S + 64.7M$	$63.4S + 52.1M$	$62.9S + 43.8M$
$s = 5$	precomp.		$3S + 3M$	$5S + 11M$	$5S + 27M$
	$\ell = 160$		$33.0S + 46.4M$	$31.9S + 36.0M$	$31.1S + 30.0M$
	$\ell = 192$		$39.7S + 55.8M$	$39.4S + 43.1M$	$38.4S + 35.7M$
	$\ell = 256$		$51.8S + 73.7M$	$51.4S + 56.9M$	$50.5S + 47.1M$
$s = 6$	precomp.		$2S + 2M$	$6S + 10M$	$6S + 26M$
	$\ell = 160$		$29.4S + 50.4M$	$29.0S + 38.8M$	$27.8S + 31.9M$
	$\ell = 192$		$32.0S + 59.6M$	$32.2S + 45.9M$	$31.7S + 37.7M$
	$\ell = 256$		$45.0S + 79.7M$	$45.7S + 60.9M$	$44.3S + 49.8M$
$s = 7$	precomp.		$1S + 1M$	$7S + 9M$	$7S + 25M$
	$\ell = 160$		$27.8S + 53.8M$	$26.9S + 40.8M$	$26.0S + 33.5M$
	$\ell = 192$		$30.1S + 64.0M$	$28.9S + 48.6M$	$28.4S + 39.5M$
	$\ell = 256$		$40.5S + 84.7M$	$39.4S + 64.1M$	$38.2S + 52.1M$

ℓ/s as expected using the new modular exponent splitting technique. Moreover, the number of multiplications performed during the multi-exponentiation slightly increases from row to row: this due to the fact that smaller digit sets have to be used to obey the space limits while the splitting parameter is increased. (Note that $s \geq 5$ cannot be used with space for only 4 dynamically precomputed elements, so the corresponding parts of the table are left empty.)

Note that the size of cache entries does not affect the statistics as reflected in the table. With severe memory constraints for the cache, $s = 2$ might be the only option. Comparing the row $s = 1$ (which describes the case of a multi-exponentiation not using a cache entry) with the row $s = 2$ shows that our technique provides for a noticeable speed-up even with just $s = 2$.

It also should be noted that our multi-exponentiation technique for old bases g_1 (Section 3.2) involves $s - 1$ divisions with remainder to perform s-fold modular exponent splitting. This starts with an ℓ-bit denominator and a divisor around $\ell - \frac{\ell}{s}$ bits; both operands will decrease by around $\frac{\ell}{s}$ in each subsequent division. Thus, the total extra cost of these modular divisions should usually be reasonably small. The typical size of results means that around ℓ bits will still suffice to store the resulting shorter exponents.

Please refer to Appendix B for certain implementation aspects. See Appendix C for a performance comparison of our technique with an immediate approach in a particular scenario.

References

1. American National Standards Institute (ANSI). The elliptic curve digital signature algorithm (ECDSA). ANSI X9.62 (1998)
2. Antipa, A., Brown, D., Gallant, R., Lambert, R., Struik, R., Vanstone, S.: Accelerated verification of ECDSA signatures. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 307–318. Springer, Heidelberg (2006)
3. Avanzi, R.M.: A note on the sliding window integer recoding and its left-to-right analogue. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 130–143. Springer, Heidelberg (2004)
4. Bernstein, D.J.: Pippenger's exponentiation algorithm. Draft (2002), `http://cr.yp.to/papers.html#pippenger`
5. Brickell, E.F., Gordon, D.M., McCurley, K.S., Wilson, D.B.: Fast exponentiation with precomputation. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 200–207. Springer, Heidelberg (1993)
6. Certicom Research. Standards for efficient cryptography – SEC 2: Recommended elliptic curve cryptography domain parameters. Version 1.0 (2000), `http://www.secg.org/`
7. Cohen, H., Ono, T., Miyaji, A.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
8. de Rooij, P.: Efficient exponentiation using precomputation and vector addition chains. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 389–399. Springer, Heidelberg (1995)
9. Gordon, D.M.: A survey of fast exponentiation methods. Journal of Algorithms 27, 129–146 (1998)
10. Institute of Electrical and Electronics Engineers (IEEE). IEEE standard specifications for public-key cryptography. IEEE Std 1363-2000 (2000)
11. Khabbazian, M., and Gulliver, T. A.: A new minimal average weight representation for left-to-right point multiplication methods. Cryptology ePrint Archive Report 2004/266 (2004), `http://eprint.iacr.org/`
12. Lim, C.H., Lee, P.J.: More flexible exponentiation with precomputation. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 95–107. Springer, Heidelberg (1994)
13. Miyaji, A., Ono, T., Cohen, H.: Efficient elliptic curve exponentiation. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 282–290. Springer, Heidelberg (1997)
14. Möller, B.: Algorithms for multi-exponentiation. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 165–180. Springer, Heidelberg (2001)
15. Möller, B.: Improved techniques for fast exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
16. Möller, B.: Fractional windows revisited: Improved signed-digit representations for efficient exponentiation. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 137–153. Springer, Heidelberg (2005)
17. Muir, J.A., Stinson, D.R.: New minimal weight representations for left-to-right window methods. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 366–383. Springer, Heidelberg (2005)
18. Muir, J.A., Stinson, D.R.: Minimality and other properties of the width-w nonadjacent form. Mathematics of Computation 75, 369–384 (2006)

19. National Institute of Standards and Technology (NIST). Digital Signature Standard (DSS). FIPS PUB 186-2 (2000)
20. Okeya, K., Schmidt-Samoa, K., Spahn, C., Takagi, T.: Signed binary representations revisited. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 123–139. Springer, Heidelberg (2004)
21. Pippenger, N.: The minimum number of edges in graphs with prescribed paths. Mathematical Systems Theory 12, 325–346 (1979)
22. Pippenger, N.: On the evaluation of powers and monomials. SIAM Journal on Computing 9, 230–250 (1980)
23. Schmidt-Samoa, K., Semay, O., Takagi, T.: Analysis of fractional window recoding methods and their application to elliptic curve cryptosystems. IEEE Transactions on Computers 55, 48–57 (2006)
24. Struik, M., Brown, D.R., Vanstone, S.A., Gallant, R.P., Antipa, A., Lambert, R.J.: Accelerated verification of digital signatures and public keys. United States Patent Application Publication US 2007/0064932 A1 (2007)

A An Example

As an illustrative toy example, let us apply our new technique to multi-exponentiations $g_1^d \cdot g_2^e$ with exponents of size at most $\ell = 18$ bits. To keep the example simple, we use unsigned-digit (instead of signed-digit) exponent representations. Let the digit sets for the fixed base be $B_2 = \{1, 3, 5, 7, 0\}$. For radix-2 and modular exponent splitting, we use splitting parameter $s = 3$. Thus, g_2 is replaced a priori by three fixed bases g_2, g_3, g_4 where $g_3 = g_2^{2^6}, g_4 = g_2^{2^{12}}$. Accordingly, we precompute the powers $(g_2, g_2^3, g_2^5, g_2^7,\ g_3, g_3^3, g_3^5, g_3^7,\ g_4, g_4^3, g_4^5, g_4^7)$ and save this data in ROM. We consider an environment with a limited amount of fast read/write memory and assume that we have only space to store 8 powers of the variable base g_1. Hence, we can choose digit set $B_1 = \{1, 3, \ldots, 15, 0\}$ for exponentiations with a new base (Section 3.1) and digit sets $B_1 = \{1, 3, 0\}, B_{G_1} = B_{G_2} = \{1, 3, 5, 0\}$ for exponentiations with an old base (Section 3.2).

Multi-exponentiation for a New Base g_1. Let us now consider the computation of $g_1^d \cdot g_2^e$ for $d = 205802 = (110010001111101010)_2$ and $e = 245153 = (111011110110100001)_2$ where g_1 is a new base, i.e. no cached precomputed data based on g_1 is available. Before the actual multi-exponentiation, we compute the powers $(g_1, g_1^3, \ldots, g_1^{15})$ and save these in fast read/write memory. Encoding $e_1 := d$ using B_1 yields

$$e_1 = (3, 0, 0, 1, 0, 0, 0, 0, 0, 0, 15, 0, 0, 5, 0, 1, 0)_2.$$

Encoding e using B_2 and then splitting into three parts e_2, e_3, e_4 yields

$$\begin{aligned} e_4 &= \quad (7, 0, 0, 0)_2, \\ e_3 &= (7, 0, 0, 5, 0, 0)_2, \\ e_2 &= (5, 0, 0, 0, 0, 1)_2. \end{aligned}$$

The following table shows what happens while performing the multi-exponentiation $\prod_{i=1}^{4} g_i^{e_i}$ as described in Section 3.1, based on interleaved multi-exponentiation as explained in Section 2.1:

j	A	Cache entry (so far)
17	1	(g_1)
16	g_1^3	
13	$(g_1^3)^{2^3} g_1 = g_1^{25}$	
12	$(g_1^{25})^2 = g_1^{50}$	$\big(g_1, (50, g_1^{50})\big)$
6	$(g_1^{50})^{2^6} g_1^{15} = g_1^{3215}$	$\big(g_1, (3215, g_1^{3215}), (50, g_1^{50})\big)$
5	$(g_1^{3215})^2 g_2^5 g_3^7 = g_1^{6430} g_2^5 g_3^7$	
3	$(g_1^{6430} g_2^5 g_3^7)^{2^2} g_1^5 g_4^7 = g_1^{25725} g_2^{20} g_3^{28} g_4^7$	
2	$(g_1^{25725} g_2^{20} g_3^{28} g_4^7)^2 g_3^5 = g_1^{51450} g_2^{40} g_3^{61} g_4^{14}$	
1	$(g_1^{51450} g_2^{40} g_3^{61} g_4^{14})^2 g_1 = g_1^{102901} g_2^{80} g_3^{122} g_4^{28}$	
0	$(g_1^{102901} g_2^{80} g_3^{122} g_4^{28})^2 g_2 = g_1^{205802} g_2^{161} g_3^{244} g_4^{56}$	

As we can see here, until and including round $j = 6$, the variable A contains no powers of bases other than g_1. Intermediate powers of g_1 for caching are available at the points $j = 12$ and $j = 6$ of the computation.

Multi-exponentiation for an Old Base g_1. Let us compute $g_1^d \cdot g_2^e$ for $d = 73660 = (10001111110111100)_2$, $e = 236424 = (111001101110001000)_2$ where g_1 is an old base for which the cache entry $(g_1, (\lambda_1 = 3215, G_1 = g_1^{3215}), (\lambda_2 = 50, G_2 = g_1^{50}))$ as created above is available. First, the powers $(g_1, g_1^3,\ G_1, G_1^3, G_1^5,\ G_2, G_2^3, G_2^5)$ are precomputed and stored in fast read/write memory. Next, we perform modular exponent splitting as described in Section 3.2:

$$\begin{aligned}
&d_0 = d = 73660,\\
&E_1 = \left\lfloor \tfrac{d_0}{\lambda_1} \right\rfloor = 22 \text{ and } d_1 = d_0 - E_1\lambda_1 = 2930,\\
&E_2 = \left\lfloor \tfrac{d_1}{\lambda_2} \right\rfloor = 58 \text{ and } d_2 = d_1 - E_2\lambda_2 = 30,\\
&E_3 = d_2 = 30
\end{aligned}$$

Encoding E_1, E_2 and E_3 using B_{G_1}, B_{G_2} and B_1 yields

$$\begin{aligned}
E_1 &= (10110)_2 = (5, 1, 0)_2,\\
E_2 &= (111010)_2 = (3, 0, 0, 5, 0)_2,\\
E_3 &= (11110)_2 = (3, 0, 3, 0)_2.
\end{aligned}$$

By encoding e using B_2 and then splitting into 3 parts e_2, e_3, e_4 (using radix-2 exponent splitting), we obtain

$$\begin{aligned}
e_4 &= (7, 0, 0, 0)_2,\\
e_3 &= (3, 0, 0, 0, 7, 0)_2,\\
e_2 &= (1, 0, 0, 0)_2.
\end{aligned}$$

The table below shows what happens in the interleaved multi-exponentiation to compute $G_1^{E_1} G_2^{E_2} g_1^{E_3} g_2^{e_2} g_3^{e_3} g_4^{e_4}$:

j	A
5	g_3^3
4	$(g_3^3)^2 G_2^3 = G_2^3 g_3^6$
3	$(G_2^3 g_3^6)^2 g_1^3 g_2 g_4^7 = G_2^6 g_1^3 g_2 g_3^{12} g_4^7$
2	$(G_2^6 g_1^3 g_2 g_3^{12} g_4^7)^2 G_1^5 = G_1^5 G_2^{12} g_1^6 g_2^2 g_3^{24} g_4^{14}$
1	$(G_1^5 G_2^{12} g_1^6 g_2^2 g_3^{24} g_4^{14})^2 G_1 G_2^5 g_1^3 g_3^7 = G_1^{11} G_2^{29} g_1^{15} g_2^4 g_3^{55} g_4^{28}$
0	$(G_1^{11} G_2^{29} g_1^{15} g_2^4 g_3^{55} g_4^{28})^2 = G_1^{22} G_2^{58} g_1^{30} g_2^8 g_3^{110} g_4^{56}$

B Implementation Aspects

On-The-Fly Signed-Digit Conversions. In our descriptions of multi-exponentiation algorithms, we have employed radix-2 representations of exponents by referring to their individual digits. However, this by no means is meant to imply that these digits need to be explicitly obtained and stored in advance, which would be quite inconvenient if memory is scarce. Left-to-right signed fractional window representations [11,23,16] are very convenient for our purposes since (for any given maximum digit value m) there is a finite-state machine that transforms the binary representation into the corresponding signed-digit representation. As the name suggests, this conversion machine starts at the most significant digit ("left") and continues towards the least significant digit ("right"). Since interleaved multi-exponentiation is a left-to-right technique as well, this often means that the signed digits can be obtained on the fly.

To make this work with radix-2 exponent splitting, we need to add an additional first left-to-right pass through the binary representation. This is essentially a dry run of the signed fractional window conversion, used to determine the first binary digits that will affect each of the segments of the signed-digit representation. For s-fold radix-2 exponent splitting, such a dry run can be used to initialize each of s finite-state machines, which afterwards can be used to obtain the digits of the individual segments (exactly as in the case of the on-the-fly conversion using just a single such machine that we would use in the case without splitting).

A simpler alternative would be to first split the binary representation, and then generate the signed-digit representations individually. This could be done truly on the fly, i.e., without the additional left-to-right pass. However, this strategy often will increase the total weight of the resulting representations [15], so the two-pass technique usually should lead to better performance.

Variants of the Signed Fractional Window Representation. In our performance estimates in Section 4, we optimistically assumed that besides ROM and fast read/write memory, there is another kind of memory that we can use for the cache. This is an assumption that we made for simplicity, but which is not necessary. In fact we may use some of the fast read/write memory for a small cache without completely losing this memory for precomputed powers of g_1.

This can be achieved by observing that we may modify the parameter m for the left-to-right signed fractional window representation *while performing the conversion.* Thus, in the algorithm from Section 3.1, provided that $m \geq 2s-1$, we may initially use some maximum-size digit set $B^{\pm}(m) = \{\pm 1, \pm 3, \ldots, \pm m, 0\}$ for signed digits $b_{1,\ell}$ down to $b_{1,L_1+\cdots+L_{s-1}}$, then cache the current group element $g_1^{(b_{1,\ell},\ldots,b_{1,L_1+\cdots+L_{s-1}})_2}$ in the memory space that so far held g_1^m, and then use the smaller digit set $B^{\pm}(m-2)$ for subsequent digits $b_{1,L_1+\cdots+L_{s-1}-1}$ down to $b_{1,L_1+\cdots+L_{s-2}}$. Continuing in this fashion, we eventually give up digits $\pm m$, $\pm(m-2)$, $\ldots$, $\pm\big(m-2(s-1)\big)$.

C Performance Comparison

This appendix demonstrates the merits of our new technique for multi-exponentiation with caching in one particular situation where very little memory is available for use as a cache. We show that our method is of advantage even under this severe restriction. We make the following assumptions:

- We look at two-fold multi-exponentiation, $g_1^{e_1} g_2^{e_2}$. Base element g_2 is fixed; base element g_1 is variable such that the current value will repeat in the directly following multi-exponentiation with probability $P_{\text{old}} = \frac{1}{2}$.
- The exponents e_1 and e_2 are uniformly random integers up to $\ell = 256$ bits.
- Storage is available for 128 fixed precomputed elements in read-only memory (derived from the fixed base g_2), and for only 2 precomputed elements in read/write memory. The latter includes input value g_1. In addition to this, we have space for variable A in the algorithm from Section 2.1, and the memory holding the exponents. (Note that typically the memory needed for the exponents is less than the memory needed for a single group element: for elliptic curve cryptography using projective coordinates over a 256-bit field, one group element takes 768 bits.)
- Different from the assumptions as used in Section 4, we have *no additional cache memory*. That is, a group element to be cached has to be kept in one of the two read/write storage units for precomputed elements.
- We use rough estimates $S = 0.7$ and $M = 1$ for the amount of time spent on each group squaring (e.g., elliptic curve point doubling) and on each group multiplication (e.g., elliptic curve point addition). (For example, when using Jacobian projective coordinates for elliptic curves over prime fields, a point doubling takes 10 or 8 field multiplications depending on the curve, and a general point addition requires 16 field multiplications [10], or 11 field multiplications in the case of "mixed coordinates" [7]. Mixed coordinates require a one-time conversion step to one of the inputs to convert it into affine coordinates, which is reasonable for precomputed values. Accordingly, $\frac{8}{11} \approx 0.73$ is one way to justify our estimate $\frac{S}{M} \approx 0.7$, although in the following we neglect the cost of the conversion.)

If (instead of applying our new caching strategy) we directly use interleaved multi-exponentiation in this situation, employing signed-digit representation as explained in Section 2.1, we can keep precomputed values $g_2, g_2^3, g_2^5, \ldots, g_2^{255}$ in read-only memory, and use read/write memory for g_1 and g_1^3, thus achieving an exponentiation cost of approximately

$$\Big(\frac{256}{4} + \frac{256}{10}\Big)M + 255S \approx 268.1$$

(or $89.5M + 254S \approx 267.3$ according to experimental simulation results) plus $1M + 1S = 1.7$ to precompute g_1^3 from g_1 when g_1 has changed from the previous computation. By assumption, this happens with probability $\frac{1}{2}$, resulting in a total estimate of $268.1 + \frac{1.7}{2} \approx 269.0$ (for the simulation: 268.2).

Our method initially performs worse than this, namely, in the case with a new base (Section 3.1). Here, the read-only memory will contain $g_2, g_2^3, g_2^5, \ldots, g_2^{127}$, plus similar powers of $g_2^{2^{128}}$. The read/write memory initially is filled with precomputed elements g_1 and g_1^3. To perform the multi-exponentiation as described in Section 3.1, we use radix-2 exponent splitting for exponent e_2 to obtain partial exponent representations no longer than 129 digits. For exponent e_1, we use a signed fractional window representation variant as sketched in Appendix B, i.e., where digit set parameter m is modified within the conversion: the more significant digits can use digits set $\{\pm 1, \pm 3, 0\}$, whereas the less significant digits (digits $b_{1,127}, \ldots, b_{1,0}$) are restricted to digit set $\{\pm 1, 0\}$. This is because we no longer keep g_1^3 in memory when the method from Section 3.1 has determined a group element to be cached, thus freeing a memory location for use as cache space. The performance estimate for this multi-exponentiation is

$$\Big(\frac{128}{4} + \frac{128}{3} + \frac{256}{9}\Big)M + 255S \approx 281.6$$

(simulation: $102.7M + 253.8S \approx 280.4$) plus $1M + 1S \approx 1.7$ to precompute g_1^3 from g_1. We benefit from the extra effort put into this computation whenever the same g_1 reappears in the following multi-exponentiation. In this case, the multi-exponentiation will only take approximate effort

$$\Big(\frac{128}{3} + \frac{128}{3} + \frac{256}{9}\Big)M + 127S \approx 202.7$$

(simulation: $113.9M + 128.2S \approx 203.6$). The average cost given $P_{\text{old}} = \frac{1}{2}$ comes to 242.1 (simulation: 242.0). Thus, our method provides an average 10 percent performance improvement in this specific scenario.

Privacy Preserving Data Mining within Anonymous Credential Systems

Aggelos Kiayias[1,⋆], Shouhuai Xu[2,⋆⋆], and Moti Yung[3]

[1] Computer Science and Engineering, University of Connecticut
Storrs, CT, USA
aggelos@cse.uconn.edu
[2] University of Texas, San Antonio, TX, USA
shxu@cs.utsa.edu
[3] Google Inc. and Computer Science, Columbia University
New York, NY, USA
moti@cs.columbia.edu

Abstract. Regular (non-private) data mining can be applied to manage and utilize accumulated transaction data. For example, the accumulated relative service time per user per month can be calculated given individual transaction data from which the user compliance with a service agreement can be determined and possibly billing can be processed. Nevertheless, due to user privacy concerns, cryptographic research developed transactions based on unlinkable anonymous credentials. Given the nature of anonymous credentials the ease of managing accumulated data (e.g., per user) is lost. To restore the possibility of management and accumulation of data it seems that a suitable form of privacy preserving data mining is needed. Indeed, privacy preserving data mining methods have been suggested for various protocols and interactions where individual data can be contributed in an encrypted form, but not within the context of anonymous credentials. Given our motivation we suggest a new notion of performing "privacy preserving data mining within the context of anonymous cryptographic credential systems," so as to protect both the privacy of individually contributed data and the identity of their sources while revealing only what is needed. To instantiate our approach we focus on a primitive we call "data mining group signatures" (DMGS), where it is possible for a set of authorities to employ distributed quorum control for conducting privacy preserving data mining operations on a batch of transactions while preserving maximum possible anonymity. We define and model the new primitive and its security goals, we then present a construction and finally show its privacy and security properties. Along the way we build a methodology that safely combines multi-server protocols as sub-procedures in a more general setting.

⋆ Research partly supported by NSF Career award CNS-0447808 and NSF SGER Grant 0751095.

⋆⋆ Research partly supported by NSF Grant IIS-0524612.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 57–76, 2008.

1 Introduction

Private (i.e., anonymous) transactions protect civil liberties. Moreover, they help service providers adhere to regulations as well as reduce potential liability. It is expected that anonymous transactions in the future will be based on notions like electronic cash [8,11], anonymous credentials [10,9], and group signatures [12]. For example, recently, it has even been proposed to integrate anonymous credentials into the new security architecture of the next generation Internet that is currently being investigated (cf., [25,31]); the intent is to aid in preventing network abuses such as Distributed Denial-of-Service (DDoS) attacks while protecting the privacy of the honest users.

However, what we claim here is that the impact of privacy on the management aspect of anonymous transaction systems, namely how to properly manage and utilize the anonymous transaction data while retaining user privacy, is not well understood, and has not been investigated sufficiently. This is true even though deanonymization of transactions has been identified as an important requirement for anonymous transaction systems (for example, group signatures (e.g., [2,3]) enable the deanonymization of a certain transaction, traceable signatures [21] enable the identification of all transactions of a certain user, fair blind signatures [30] enable the "unblinding" of the signing protocol, offline e-cash [11] enable the deanonymization of the transaction source in case of double spending or even n-time spending in the case of [7], and unclonable identification [13,6] prohibits over-usage within time periods).

Still, deanonymization is only one limited aspect of what administration of an anonymous system is interested in. As a matter of fact, in none of the anonymous transaction systems mentioned above it is possible to extract, for example, information about usage statistics within a time period without revoking the anonymity of the transactions first. Such management operations are crucial in assuring that users perform within usage bounds (that are "percentage-wise" within time periods, e.g., no more than 10% of transactions in a batch), and in enabling billing users based on their (e.g.) monthly usage etc. In general, the administration may be interested in performing various management operations on the transaction data, and in revealing different aspects of the data distribution and perhaps imposing deanonymization when the distribution gets skewed for some of the transaction originators.

The goal of this paper is to initiate the study of privacy in data available from anonymous credential transaction systems. To this end we present the notion of "data mining group signatures," which enable administrators to write small "code snippets" that will be carried out distributedly by a set of mining authorities and will compute (in a distributed manner over a protocol procedures library) the desired output of a mining operation in a privacy preserving manner.

Other related work. Previous studies on privacy preserving data mining (see, e.g., [24,1,16,22,20]), focused on processing multiple private datasets, which are in the form of plaintext but should not be mutually disclosed. The difference here

is that the data being processed are the "identification tags," which are implicitly collected from anonymous transactions and are in the form of ciphertexts.

1.1 Our Results

Basic contributions. Let us reiterate that our first contribution is identifying the need of privacy preserving data mining in the context of anonymous transactions. We focus on designing a transaction system that enables distributed operations on the scrambled identity tags that accompany anonymous transactions. The notion allows a group of system servers to perform data mining with trusted quorum control: as long as the majority of servers is honest, only the required mining operations are performed and system privacy is preserved. This general notion and its applications for operation and control are then modeled in the context of group signatures, a very basic, general and long-lived anonymous credential system.

In addition we propose a model for performing and sequentially combining a suite of multi-server secure computational tasks (threshold cryptography, and round table protocols). The composition preserves the security correctness and robustness of the functionality when performed by a single trusted (black-box) entity. This leads to a proof of security and correctness against a static minority-controlling adversary. While earlier works considered such proofs of protocols in isolation, a proof for a dynamic combination of protocols among a suite of protocol procedures was not available in the literature.

Implications. Our motivation gives rise to two exemplifying usage characteristics called "usage histogram" and "blinded usage histogram with outlier detection," which are implemented efficiently within a data mining group signature scheme without further privacy exposures. Usage histograms can be computed in order to aggregate transactions per user. A typical application of a usage histogram is statistics or billing. While there is no linking to the original transactions, the data mining system in this case is capable of obtaining a histogram that shows the number of transactions per user. Compared to usage histograms, blinded usage histograms can better protect the users' anonymity because there is no identity information in the histograms — the histogram is extracted but the names corresponding to each column are blinded. A typical application may be in detection of misbehavior in violation of a service agreement by observing the usage histograms and isolating columns that are, say, too high. The data mining system can then obtain unmasked identities of exactly those users that correspond to the selected columns.

Applications. We stress that using our methodology one can write other little abstract programs (i.e., snippets) that can use the suite of basic procedures and be executed by the mining servers to run any mining operation. Nevertheless, the exemplifying histograms and blinded histograms by themselves can already find specific applications such as the following.

* For each period of time (e.g., every month), the authorities can extract the usage counts of each user's data mining group signatures. This allows the authority to detect any significant deviation from a user's normal usage, which may cause further investigations. This utility can be used as a way to detect misbehaving parties or simply maintain statistics about the system (without actually linking a user to the actual transactions).
* Given the network log of packets during a packet flooding attack, where each packet has an associated data mining group signature (that is validated for packets to be routed) as suggested in one of the next generation Internet security architecture proposals, the authority (i.e., a set of parties running the distributed cryptosystems presented in this paper) can identify the senders that send packets more than a threshold because they are potentially the attackers, whereas the senders who send a small number of packets remain anonymous. Note that signing and verifying need only be applied to fractions of packets to preserve efficiency.
* In business management, usage histograms can be utilized for operations like *billing* (as mentioned above). This can also be used to detect dead accounts (where there was no usage). Moreover, the histograms can also serve as a basis for developing more advanced applications such as privacy preserving decision tree mining [23] and Bayes classifier learning [32].

Discussion: Our approach vs. E-cash. One may consider employing (virtual) e-cash for usage monitoring (not as a payment but for monitoring purposes). Let us examine how one may attempt to use e-cash as a technical tool for monitoring activities. Recall that in the e-cash model every user gets coins (or an amount that can be split into coins); in our setting each participant will draw from the bank "server coins" that will be used for transactions in the next time period. When a user performs an anonymous transaction a coin is used so that the user cannot be traced and the transaction remains anonymous. This has several implications.

First, in e-cash the maximum usage has to be known a-priori. If the a-priori number differs among users, it already reveals something about their intended usage and is in violation of anonymity. For example, if the a-priori amount is used and a user needs additional coins, this reveals over-usage before the monitoring is conducted (the bank knows the identity of users). On the other hand, if the bank is prohibited from knowing the identity of users (as a remedy to the above drawback), users may draw a lot of coins beyond need (and malicious users may combine their pool of coins without initially committing to "who will use the money").

As another remedy, one may suggest that in the case of users drawing the same number of coins (which as indicated above it is already problematic), at the end of the period users who did not use all their coins will deposit back their coin (so a user that did not get any service will deposit all her coins). This would require all users to act in a period even if they have no intention whatsoever of using the service; forcing subscribers to take action even if they are not active in a period is a problem from usability/system-management and may be unacceptable from

a user point of view (e.g., user may be allowed to be off-line for a period); on the other hand, not forcing all users to be active may reveal usage intention in violation of anonymity. Regarding depositing the unused coins, it should be also assured that a depositing user is indeed depositing the coins she withdrew (without combining them with other users) while also possibly preserving the anonymity of the user who is depositing back her coins. Otherwise, misbehaving users combining their states can bias the anonymous distribution, so that if one user over-used the system and the second user under-used the system, they can negotiate coin exchange so they show that they both are behaving well. If a user shows the unspent coins under his name, this is not a blinded histogram any more. Additionally, even the case of a named histogram, which may be used to detect over-spending and under-spending at the same time, cannot be achieved because if a user under-spent, she may choose not to show all her remaining coins and thus cheat the system simply this way.

The above limitations are caused by the fact that monitoring via long-lived credentials as we do here can be used in itself for billing and is a natural "pay per use" method, whereas e-cash has to be drawn first and represents essentially a "debit card" like instrument.

Moreover, as argued above, the coins themselves are anonymous. Thus, to allow usage histograms based on names, the coins have to be de-anonymized, (i.e., escrowed coins – a mechanism that exists). But once de-anonymized, the coins are still associated with the transactions where they were spent (where the "payment" took place). This jeopardizes the anonymity of the transactions. Therefore, we do not see that e-cash helps in such anonymous monitoring of usage that is independent of the transactions without requiring another layer of anonymization.

In summary, the e-cash method as a tool to solve privacy-preserving usage monitoring in the case of anonymous credentials, does not seem to solve the problem, though superficially it seems related. Its inadequacy is based on various objections, some based on attacks on the system and misuses, and some based on unsuitability to the target goal of usage monitoring. Finally we recall that, in fact, e-cash is a form of anonymous credentials that the mechanisms we propose here can augment to accommodate some form of monitoring if suitably modified; but we chose to demonstrate our approach based on group signatures whose long-lived credential nature is particularly suited to our objectives.

Organization: The rest of the paper is organized as follows. In section 2 we discuss our methodology for composing multiserver protocols for ciphertext computations. In section 3 we present a model for data mining group signatures (DMGS). In section 4 we present a concrete DMGS scheme. In section 5 we conclude the paper.

2 MultiServer Protocols and Their Composition

The section presents our methodology of composing a suite of distributed multiserver protocols for specialized computation over ciphertexts. This building

block will be used to construct the data-mining group signature scheme in section 4. The model of allowing arbitrary sequential composition of such protocols is novel; for ease of presentation and simplification we will employ the random oracle model in our construction and assume a static adversary throughout. The protocols will be carried out by a set of servers $\mathcal{M}_1, \ldots, \mathcal{M}_m$.

Distributed Multi-server Protocols. An m-server protocol execution is based on an interactive program P that is run by all servers and, provided that $m > 1$, an adversarial program $\mathcal{A}$ who controls at most $t-1$ of the servers subject to the constraint $m \geq 2t-1$; the execution has the following characteristics:

- *Participants.* The servers $\mathcal{M}_1, \ldots, \mathcal{M}_m$ and the adversary $\mathcal{A}$. At the start of the protocol the adversary selects a set of up to $t-1$ servers to corrupt such that $m \geq 2t-1$.
- *Input and Output.* Each participating server is given private and public input. The input to each server includes the number of servers m (it is assumed that the protocol P operates on any given number of servers). At the end of the computation each server running program P will produce private output as well as a public output that should be equal among all honest servers. The private input of corrupted servers as well as the public input is given to the adversary at the start of the protocol. The public input includes the security parameter 1^κ (and note that all parties are polynomial-time bounded in κ).
- *Communication Model.* We assume that the communication is synchronous and that the protocol execution proceeds in rounds. In each round the program of each server using its current state and history of communication up and including the previous round produces two types of messages to be delivered to other servers; the first type is a point-to-point private message that is delivered privately to the intended recipient. This type of transmission models secure authenticated point to point channels between a pair of servers. The second type of message is a broadcast message that will be delivered to all servers at the beginning of the next round. At each round a server produces private messages for all other servers as well as a public broadcast message.
- *Adversarial Operation.* At each round the adversary is activated last, after the honest participants have submitted their messages. Based on the private messages directed to the corrupted parties, the public broadcast messages as well as all information available from previous rounds to the corrupted parties, the adversary decides the public and private messages that will be delivered from the servers under its control, i.e., the corrupted servers in this round.

A **suite** of m-server protocols $\mathtt{Suite} = \langle \mathtt{PROT}_1, \ldots, \mathtt{PROT}_s \rangle$ is a set of protocols that can be executed sequentially and use a joint state. In particular after an initialization protocol is performed (by convention $\mathtt{PROT}_1$) subsequent protocols executions can be sequentially composed in an arbitrary fashion and they will all employ the same state. The private input of the server in each protocol will

be the current state of the server. The public input to the set of servers will be provided externally. We will use a special notation $\langle \mathsf{Prot}_1, \ldots, \mathsf{Prot}_s \rangle$ to denote the single server versions of the programs of the protocols $\langle \mathtt{PROT}_1, \ldots, \mathtt{PROT}_s \rangle$ (recall that each $\mathtt{PROT}_\ell$ is defined for any number of servers).

An execution of an m-server protocol suite $\mathtt{Suite}$ denoted by $\mathcal{E}_{\mathcal{A}}^{\mathtt{Suite}}(1^\kappa, 1^m)$ where $\mathcal{A}$ is the adversary, is a simulation of the program of $\mathcal{A}$ that proceeds as follows: (i) First the adversary $\mathcal{A}$ is executed with inputs $1^m, 1^\kappa$ and it selects a set of at most $t-1$ servers to corrupt subject to the constraint $m \geq 2t-1$. (ii) The initialization protocol $\mathtt{Prot}_1$ is executed as described above with the adversary participating on behalf of the corrupted servers; note that this protocol requires no private inputs for any of the servers and its public input is $1^\kappa, m$; the private outputs of honest servers are maintained as an internal state of the execution that is inaccessible to the adversary. (iii) Subsequently, the adversary $\mathcal{A}$ may provide a public input and ask the honest servers to execute together with the corrupted servers under the adversary's control any of the protocols in the suite. This step can be repeated sequentially as many times as the adversary commands. (iv) The adversary $\mathcal{A}$ may terminate the execution at any time outputing a single bit which is also the output of the execution $\mathcal{E}$.

Definition 1. *A suite of m-server protocols $\mathtt{Suite} = \langle \mathtt{PROT}_1, \ldots, \mathtt{PROT}_s \rangle$ is called* t**-distribution-safe** *if for all adversaries $\mathcal{A}$ corrupting less than t servers, it holds that there exists an expected polynomial-time simulator $\mathcal{S}$ such that for all $m \geq 2t-1$,*

$$|\mathbf{Prob}[\mathcal{E}_{\mathcal{A}}^{\mathtt{Suite}}(1^\kappa, 1^m) = 1] - \mathbf{Prob}[\mathcal{S}^{\mathsf{Prot}_1, \ldots, \mathsf{Prot}_s}(1^\kappa) = 1]| = \mathsf{negl}(\kappa)$$

The intuition behind this definition is that an adversary that controls at most $t-1$ servers is incapable of gaining any advantage due to server corruption while executing an arbitrary sequential composition of the protocols in the suite. This is argued by the fact that the adversary's knowledge gain can be simulated by a sequential execution of the same set of protocols with a single trusted server (represented by the set of oracles available to the simulator $\mathcal{S}$). Given that anything the adversary can compute in the corrupted server setting, it can also compute while interacting with a single honest server we conclude that the protocol suite is "distribution-safe." Note that distribution-safety as a property suggests that a single-server functionality is distributed to a set of servers in a manner that any correctness or security property that the protocol suite may satisfy is preserved. Distribution safety does not impose by itself any correctness or security guarrantee on the protocol suite (and these will be argued separately).

We next define a protocol suite that relates to ElGamal encryption and we will take advantage of in our construction. We first describe the encryption function itself that is loosely based on [29] and then we describe the protocol suite that we will need. Let $\langle G_1 \rangle$ be a group of prime order q; the public key includes G_1, G_2, H, q where $G_2, H \in \langle G_1 \rangle$. The secret key is the value $w = \log_{G_1} H$. Given a plaintext $M \in \mathbb{Z}_q$ the encryption is the tuple $\langle U_1, U_2, C \rangle = \langle G_1^r, G_2^r, H^r G^M \rangle$ where $r \leftarrow_R \mathbb{Z}_q$ accompanied with a non-interactive proof of knowledge for the

statement $\mathrm{PK}(\rho : U_1 = G_1^\rho \wedge U_2 = G_2^\rho)$; this is a proof of equality of discrete-logs that can be made non-interactive following the Fiat-Shamir heuristic [15]. Overall the ciphertext will have the form $\Omega = (U_1, U_2, C, \pi)$ with π standing for the non-interactive proof of knowledge. Occasionally, we may use the notation Ω° to denote (U_1, U_2, C) and call this the "reduced ciphertext." This would be useful in contexts where it is certain that the ciphertext is valid (i.e., the U_1, U_2 are properly formed).

We note that we do not require the actual recovery of M (thus encrypting G^M does not hurt the efficiency of decryption); alternatively one can think of the size of the plaintext space as polynomial in κ (this would be indeed the case in our setting) and thus the recovery of M is possible through exhaustive search (or even a baby-step giant-step strategy). Following [29] one can show that the above cryptosystem is IND-CCA2 in the random oracle model assuming the Decisional Diffie Hellman assumption.

We proceed next to define a protocol suite for the encryption scheme defined above that is parameterized by two hash functions $\mathcal{H}, \mathcal{H}'$. Each server maintains a set $QUAL$ that contains the set of properly acting servers in a sequence of an execution. The way that the protocols in the suite maintain $QUAL$ will guarantee that all honest servers maintain the same set and in all cases $|QUAL| \geq t$. The protocols for the servers $\mathcal{M}_1, \ldots, \mathcal{M}_m$ are as follows:

- `ParGen` is an m-server protocol with public output defined by the probabilistic function f, where $f(1^\kappa)$ is a tuple that includes the description of a group that contains G_1 as well as the κ-bit prime q which is the order of G_1. We assume that this the group is selected from a predetermined table (that contains one entry for each κ) and thus `ParGen` is non-interactive.
- `ExpGen` is an m-server protocol that using the group description of $\langle G_1 \rangle$ and the parameters t, m, it enables the i-th server to compute a public output $(H, H_1, \ldots, H_{t-1})$ as well as the private output w_i where $H = H_0 = G_1^w$ is a random element of $\langle G_1 \rangle$ and it holds that $w \overset{(t,m')}{\longleftrightarrow} (w_{i_1}, \ldots, w_{i_{m'}})$ where $QUAL = \{i_1, \ldots, i_{m'}\} \subseteq \{1, \ldots, m\}$ as well as $H_0 H_1^{i_\ell} \ldots H_{t-1}^{i_\ell^{t-1}} = G^{w_{i_\ell}}$ for $\ell = 1, \ldots, m'$. The notation $w \overset{(t,m)}{\longleftrightarrow} (w_1, \ldots, w_m)$ means that $w_1, \ldots, w_m$ is a secret-sharing of w (cf. [28]) so that using any t out of the m shares it is possible to reconstruct w but any less than t shares reveal no information about w.

  This protocol can be realized by the distributed key generation `DKG` protocol of [18] which builds on [14,27].

  We note that the `DKG` protocol relies on Pedersen commitments which also require a value $F \in \langle G_1 \rangle$ with unknown discrete-logarithm. Such value can be calculated by having each server computing $F = \mathcal{H}(\tau)$ where τ is a fixed string known to all parties (and is unique for each invocation of the system); we assume that the range of $\mathcal{H}$ can be mapped to $\langle G_1 \rangle$. Note that the protocol fails if $F = 1$ (a negligible probability event).

– `PkGen2`. This is an m-server protocol to compute the value G_2. The calculation of G_2 can be done in the same way as the value F given above but using the hash function $\mathcal{H}'$ instead.

* `Init` : the sequential composition of the protocols (`ParGen`, `ExpGen`, `PkGen2`) in this order constitutes the initialization of the protocol suite. Subsequent protocol executions employ the parameters generated by this execution.

– `ExpRecon` is an m-server protocol that on input $V \in \langle G_1 \rangle$ it produces the public output V^w where $w \overset{(t,m')}{\longleftrightarrow} (w_{i_1}, \ldots, w_{i_{m'}})$ is the secret-sharing of the secret-key (committed to $H = G^w$) that is held by the $QUAL = \{i_1, \ldots, i_{m'}\}$ subset of the m servers.

The protocol is realized as follows: The i-th server broadcasts $V_i = V^{w_i}$ as well as a non-interactive proof of knowledge of the statement $\mathrm{PK}(\alpha : V_i = V^\alpha \wedge H_0 H_1^j \ldots H_t^{j^t} = G^\alpha)$; this is a proof of equality of discrete-logarithms that is made non-interactive using the hash function $\mathcal{H}'$. Upon receiving the values $V_{i_1}, \ldots, V_{i_{m'}}$ accompanied by the NIZK's $\pi_{i_1}, \ldots, \pi_{i_{m'}}$, each server finds t values V_i for which the proofs are valid, say $\Lambda = \langle i_1, \ldots, i_t \rangle$ and computes $\prod_{l \in \Lambda} V_{i_l}^{\lambda_l^\Lambda}$ where $\lambda_1^\Lambda, \ldots, \lambda_t^\Lambda$ are the Lagrange coefficients that satisfy $\sum_{l \in \Lambda} \lambda_l^\Lambda \cdot p(i_l) = p(0)$ for all polynomials p of degree less than t in $\mathbb{Z}_q$.

If a server finds that the proof π_i is not valid, it removes server i from the set $QUAL$.

– `DEC` is an m-server protocol that on input a ciphertext (U_1, U_2, C, π) it returns the decryption G^M of the ciphertext or the value $\perp$ to stand for failure. Specifically, given a ciphertext, the i-th server checks whether the proof π is valid; in case the test fails the server outputs $\perp$. Otherwise, the servers in $QUAL$ execute the `ExpGen` protocol on input U_1 to compute $U_1^w = H^r$ and subsequently decrypt C by returning C/H^r as public output.

– `MIX` is an m-server protocol that on input a sequence of ciphertexts $\Omega_1, \ldots, \Omega_n$ it outputs a sequence of ciphertexts $\Omega'_1, \ldots, \Omega'_n$ such that if $\langle L_1, \ldots, L_n \rangle$ is the vector of plaintexts of the given ciphertexts, the output of the protocol is a vector of ciphertexts whose plaintexts are as follows $\langle L_{p(1)}, \ldots, L_{p(n)} \rangle$ for some randomly selected permutation p.

The `MIX` protocol follows a roundtable format: according to a schedule, each server reencrypts and shuffles a vector of ciphertexts $\langle \Omega_1^\circ, \ldots, \Omega_n^\circ \rangle$; then it broadcasts the shuffled list together with a proof of a correct shuffle that ensures that all plaintext values have been retained. Note that each server acts on the output of a previous server according to the schedule. If a server is found to produce an incorrect shuffle, the protocol restarts with the misbehaving server removed from $QUAL$. There are a number of protocols that are suitable for our setting e.g., [17,26,19]. Below we describe our `MIX` based on a shuffling protocol that builds on [19].

The first server $\mathcal{M}_j$ in the schedule will check all NIZK proofs that accompany the input vector of ciphertexts. It will then operate on the reduced ciphertexts $\Omega_1^\circ, \ldots, \Omega_n^\circ$. We modify the shuffle protocol of [19] as follows: in a first stage each server will broadcast a commitment to the permutation it

will use as well as an NIZK that ensures the commitment is properly formed (broadcasting this commitment is also part of the shuffling protocol). Once this stage is completed the servers will execute the shuffling protocol according to the schedule adhering to their original commitments and using the Fiat-Shamir heuristic with the hash function $\mathcal{H}'$ to make the proof non-interactive. The parameters for the Pedersen type of commitment as the one used in [19] can be produced by the m servers by executing the protocol `PkGen2` prior to the execution of the mixing protocol (using fixed strings derived from the τ string each time).

– `CMP` is an m-server protocol that on input two ciphertexts Ω, Ω' it returns public output 1 if and only if $\texttt{DEC}(\Omega) = \texttt{DEC}(\Omega')$ (and 0 otherwise). We notice that `CMP` as a protocol relates to a private equality test (or PET), see, e.g., [16]. However, PET is a two-party protocol where two parties wish to check whether their private values are equal or not; on the other hand in a `CMP` protocol a set of servers operate on two ciphertexts and wish to check whether the corresponding plaintexts are equal when nobody gets to know the decryption of the ciphertexts.

We present two solutions to the above `CMP` protocol problem that, depending on the network connectivity between the servers either one can be more suitable. The basic idea underlying them is the following observation: Let $\Omega = \langle U_1, U_2, C, \pi \rangle$ and $\Omega' = \langle U_1', U_2', C', \pi' \rangle$, and define $\Psi^\circ = \langle U_1/U_1', U_2/U_2', C/C' \rangle$. Observe that, assuming Ω, Ω' were valid ciphertexts encrypting $G^M, G^{M'}$ respectively, Ψ° is a valid reduced ciphertext for the value $G^{M-M'}$. It follows that if $M = M'$ the reduced ciphertext Ψ° encrypts G^0 and this property can be tested without leaking substantial information about M, M'.

Roundtable protocol for `CMP`. The first solution is suitable for settings where the servers prefer to minimize broadcasting. It includes the following stages:

1. Each server $\mathcal{M}_j$ selects a random value a_j selected from $\mathbb{Z}_q^*$ and broadcasts a commitment to a_j denoted by $\psi = C(a_j)$ as well as a NIZK proof that the commitment is well-formed.
2. Given the ciphertexts Ω, Ω', the server $\mathcal{M}_j$ where j is the smallest value in $QUAL$, "random scales" the ciphertext, an operation denoted by $\Psi^\circ \xrightarrow{\mathsf{r-s}} \Psi'^\circ$, that proceeds as follows: the server computes $\Psi'^\circ = (V_1, V_2, D) = (U_1^{a_j}, U_2^{a_j}, C^{a_j})$ where $\Psi^\circ = (U_1, U_2, C)$ and a_j is the value committed in stage 1. The server broadcasts Ψ'°. Observe that after this operation is executed by the first server the resulting ciphertext Ω'° is either an encryption of G^0 (if $M_1 = M_2$) or a valid ciphertext of the plaintext $G^{a_j(M_1 - M_2)}$. Moreover, if $M_1 \neq M_2$, then $a_j(M_1 - M_2)$ is uniformly distributed over $\mathbb{Z}_q^*$. Finally $\mathcal{M}_j$ computes a NIZK proof $\mathrm{PK}(\alpha : V_1 = U_1^\alpha \wedge V_2 = U_2^\alpha \wedge D = C^\alpha \wedge \psi = C(a_j))$ based on the hash function $\mathcal{H}'$. The next server in $QUAL$ collects (V_1, V_2, D), verifies the proof and repeats the process. If a server produces an invalid proof it is removed from $QUAL$ and the protocol is restarted.

3. After at least t of the participating servers execute step 2 (this will be guaranteed by the assumption that $m \geq 2t-1$ and the fact that the adversary controls at most $t-1$ servers), the servers enter the third stage of the protocol: if Ψ° is the final result from stage 2, the servers execute the protocol `DEC` on Ψ° (omitting the part where the proof is being checked). The servers conclude by returning "1" if the decryption of Ψ° results in 1 (i.e., $G^0 \bmod P$) and "0" otherwise.

Threshold protocol for `CMP`. The second protocol solution to `CMP` is more suitable for cases where there is a great number of servers and broadcasting is an inexpensive operation (in this setting the roundtable approach of the first solution may be inefficient). It will be broken into the following stages:

1. The servers execute `ExpGen` on group $\langle G_1 \rangle$ to produce $Z_0, Z_1, \ldots, Z_{t-1} \in \langle G_1 \rangle$; recall that this results in $Z = Z_0 = G^z$ such that $z \overset{(t,m)}{\longleftrightarrow} (z_{i_1}, \ldots, z_{i_{m'}})$ for some subset $QUAL' = \{i_1, \ldots, i_{m'}\}$ of $QUAL$. The internal state of each server contains now the share z_i.
2. The servers in $QUAL'$ as determined from the previous stage, execute three instances of the protocol `ExpRecon` on input U_1, U_2, C respectively, where $\Psi^\circ = (U_1, U_2, C)$. This results in the scaled ciphertext $\Psi'^\circ = (V_1, V_2, D) = (U_1^z, U_2^z, C^z)$.
3. The servers execute the protocol `DEC` on Ψ'° (omitting the part where the proof is being checked). The servers conclude by returning "1" if the decryption of Ψ'° results in 1 (i.e., $G^0 \bmod P$) and "0" otherwise.

We conclude the section by showing that the protocol suite we defined above is t-distribution-safe:

Theorem 1. *The suite of m-server protocols $\langle$`ParGen`, `ExpGen`, `PkGen2`, `DEC`, `MIX`, `CMP`$\rangle$ described above is t-distribution-safe assuming the discrete-logarithm assumption and that $\mathcal{H}$ is a random oracle controlled by the simulator, for $m \geq 2t-1$ servers.*

3 Data Mining Group Signatures (DMGS): Model

We now define "data mining group signatures" (DMGS), that extend the notion of group signatures with a `MINING` code snippet: a distributed algorithm that can be executed by a quorum of mining servers and will be based on the ciphertext manipulations and operations of the previous section; the code computes a given usage characteristic (the data mining objective of the system). The participants involved in the system are the users/signers, the DMGS manager (i.e., credential issuer) that is denoted by DMGM, and the data mining servers $\mathcal{M}_1, \ldots, \mathcal{M}_m$. The adversary we will assume will be t-threshold meaning that it can corrupt at most $t-1$ mining servers. While the corruptions of the mining servers will be assumed to be static in our security modeling the adversary can adaptively corrupt the group members.

Definition 2. (DMGS) *A data-mining group-signature scheme is comprised of:*

1. Setup*: This stage consists of two parts.*
   (a) $\mathsf{KeyG}_{\mathsf{DMGM}}$*: On input a security parameter* κ *and an upper bound on the number of users* n*, this probabilistic algorithm outputs the data mining group signature manager's public key* $\mathcal{Y}_{\mathsf{DMGM}}$ *(including all system parameters), the public user database, and the secret keys of all users* $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$*. The secret keys* $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ *are distributed privately to the users and the* DMGM *terminates by discarding all its random coin tosses. To each user key* sk_i *there is a corresponding public-key* pk_i *and a name* id_i *that are part of the public user database* $\{\mathsf{id}_i, \mathsf{pk}_i\}_i$ *(we may refer to this table as:* public_user_database*). The table can be accessed through a function* $\mathsf{tableLook}(\cdot)$*, that given* pk_i *returns* id_i*, the identity of the* i*-th user. Without loss of generality we will assume that* $\mathsf{id}_i = i$ *but in practice* id_i *may contain more information about the user.*
   (b) $\mathsf{KeyG}_{\mathsf{DM}}$*: this is an* m*-server protocol that with public input the parameters* $1^\kappa, t, m$*, it enables the data mining servers* $\mathcal{M}_1, \ldots, \mathcal{M}_m$ *to produce as public output the public key* $\mathcal{Y}_{\mathsf{DM}}$ *that will be attached to the public key of the system, which is denoted by* $\mathcal{Y} = \mathcal{Y}_{\mathsf{DMGM}} || \mathcal{Y}_{\mathsf{DM}}$*. At the completion of the protocol each data mining server will also return the private output* $\mathcal{S}_j$ *which will be a share of the virtual key* $\mathsf{sk}_{\mathsf{DM}}$*.*
2. Sign*: A probabilistic algorithm that given the system public key* $\mathcal{Y}$*, a user's secret key* sk_i*, and a message* M*, it outputs a signature for the message* M*. We write* $\mathtt{SIGN}(\mathcal{Y}, \mathsf{sk}_i, M)$ *to denote the application of the signing algorithm. A signature* δ *produced by the* SIGN *algorithm contains a mining tag denoted by* mt_δ*.*
3. Verify*: An algorithm for establishing the validity of a signature on a message with respect to a system public key* $\mathcal{Y}$*. Notice that* $\mathtt{VERIFY}(\mathcal{Y}, M, \delta) \in \{\text{TRUE}, \text{FALSE}\}$*.*
4. OPEN *is an* m*-server protocol that given a signature* δ*, it enables the mining servers* $\mathcal{S}_1, \ldots, \mathcal{S}_m$ *recover a value* pk_i *that can be used to identify a user from the public user database, or the value* $\perp$*. When it will be clear from the context what servers are participating in the execution we will denote the output of the protocol simply by* $\mathtt{OPEN}(\delta)$*.*
5. MINING*: This is an* m*-server protocol that enables the mining servers to collaboratively compute some application-dependent usage characteristic function* $\mathsf{usageChar} : \mathbb{N}^* \to T$ *where* T *is some arbitrary range. The protocol* MINING *will be expressed as an algorithm that is executed by each mining server locally and includes calls to the subprotocols* MIX*,* $\mathtt{tableLook}$(DEC)*, and* CMP *that operate on the mining tags of a given vector of valid signatures. Two concrete implementations will be presented in Section 4. When it will be clear from the context what servers are participating in the execution we will denote the output of the protocol by simply* $\mathtt{MINING}(\delta_1, \ldots, \delta_K)$*.*

A PPT adversary $\mathcal{A}$ for a DMGS has access to the following oracles: Setup via $\mathsf{KeyG}_{\mathsf{DMGM}}$ and $\mathsf{KeyG}_{\mathsf{DM}}$; OSign which receives a user's identity i and a message

M, returns $\mathsf{Sign}(\mathcal{Y}, \mathsf{sk}_i, M)$, and sets $\mathsf{hist}(\mathsf{Sign}) = \mathsf{hist}(\mathsf{Sign})||(i, M)$; `MINING` and `tableLook`; Corrupt which receives a group signature user's identity i and returns sk_i and sets $Corr = Corr \cup \{i\}$.

The formalization of the security properties will be performed in the setting of a *single* honest mining server. Subsequently, arguing the security of our multi-server construction will be split in two steps: first we will prove that it satisfies the properties stated below in the single server setting; then we will show that it satisfies distribution-safety as defined in the previous section, hence the advantage cannot gain significant advantage from corrupting a minority of servers.

Definition 3. (security of DMGS) *The security properties of data mining group signatures are:*

1. *Correctness: We require that a scheme possesses "signature correctness" that suggests the following probability is overwhelming:*

$$\Pr\left[\begin{array}{l}\langle \mathcal{Y}_{\mathsf{DMGM}}, \mathtt{public\_user\_database}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n\rangle \leftarrow \mathtt{KeyG}_{\mathsf{DMGM}}(1^\kappa, n);\\ \langle \mathcal{Y}_{\mathsf{DM}}, \mathcal{S}\rangle \leftarrow \mathtt{KeyG}_{\mathsf{DM}}(\mathcal{Y}_{\mathsf{DMGM}}, t = 1, m = 1); \quad \mathcal{Y} := \mathcal{Y}_{\mathsf{DMGM}}||\mathcal{Y}_{\mathsf{DM}};\\ \delta \leftarrow \mathsf{Sign}(\mathcal{Y}, \mathsf{sk}_i, M) : \text{TRUE} = \mathsf{Verify}(\mathcal{Y}, M, \delta) \wedge i = \mathsf{tableLook}(\mathtt{OPEN}(\delta))\end{array}\right]$$

and similarly for "mining correctness" the following probability is overwhelming:

$$\Pr\left[\begin{array}{l}\langle \mathcal{Y}_{\mathsf{DMGM}}, \mathtt{public\_user\_database}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n\rangle \leftarrow \mathtt{KeyG}_{\mathsf{DMGM}}(1^\kappa, n);\\ \langle \mathcal{Y}_{\mathsf{DM}}, \mathcal{S}\rangle \leftarrow \mathtt{KeyG}_{\mathsf{DM}}(\mathcal{Y}_{\mathsf{DMGM}}, t = 1, m = 1); \quad \mathcal{Y} := \mathcal{Y}_{\mathsf{DMGM}}||\mathcal{Y}_{\mathsf{DM}};\\ (\delta_1, \ldots, \delta_K) \leftarrow \mathcal{A}^{\mathsf{OSign}, \mathtt{MINING}, \mathsf{Corrupt}}(1^\kappa);\\ \mathsf{output}_1 \leftarrow \mathsf{usageChar}(\mathsf{tableLook}(\mathtt{OPEN}(\delta_1)), \ldots, \mathsf{tableLook}(\mathtt{OPEN}(\delta_K)));\\ \mathsf{output}_2 \leftarrow \mathtt{MINING}(\delta_1, \ldots, \delta_K)] : \mathsf{output}_1 = \mathsf{output}_2\end{array}\right]$$

recall that the `MINING` *protocol is a code snippet that employs the subprotocols* `DEC`, `CMP` *and* `MIX` *(as defined in section 2) that operate on the mining tags of the signatures.*

2. *Traceability: For any* PPT *adversary $\mathcal{A}$, the following probability is negligible:*

$$\Pr\left[\begin{array}{l}\langle \mathcal{Y}_{\mathsf{DMGM}}, \mathtt{public\_user\_database}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n\rangle \leftarrow \mathtt{KeyG}_{\mathsf{DMGM}}(1^\kappa, n);\\ \langle \mathsf{aux}, \mathcal{Y}_{\mathsf{DM}}, \mathcal{S}_1, \ldots, \mathcal{S}_m, m, t\rangle \leftarrow \mathcal{A}(\mathcal{Y}_{\mathsf{DMGM}}); \quad \mathcal{Y} := \mathcal{Y}_{\mathsf{DMGM}}||\mathcal{Y}_{\mathsf{DM}};\\ \langle M, \delta\rangle \leftarrow \mathcal{A}^{\mathsf{Sign}, \mathsf{Corrupt}}(\mathcal{Y}, \mathsf{aux}) \ s.t. \ (i, M) \notin \mathsf{hist}(\mathsf{Sign});\\ \text{TRUE} \leftarrow \mathtt{VERIFY}(\mathcal{Y}, M, \delta); \quad i \leftarrow \mathsf{tableLook}(\mathtt{OPEN}(\delta)) :\\ (i \notin Corr) \vee (i \notin \{1, \ldots, n\})\end{array}\right]$$

3. *Anonymity: an adversary $\mathcal{A}$ against anonymity is a* PPT *that receives the public-key $\mathcal{Y}$ of the system as well as it is allowed to corrupt any number of signers adaptively. The identifiers of the corrupted signers are maintained in a set $Corr$. Furthermore $\mathcal{A}$ interacts with three oracles* OSign, Open, Mining, *where* OSign *stands for an oracle that receives (i, M) and returns a signature on behalf of i-th signer whereas* Open *and* Mining *stand for the single (honest) server executions of the corresponding two protocols defined above.*

Consider now an oracle anonSign *with the specification that it takes as input a pair* (i, M) *but it ignores its first input (which is the user's identity) and records all its answers. We also define two oracles* anonOpen *and* anonMining. *(1)* anonOpen *takes as input a signature* δ *on* M*; if* δ *is not valid it returns* $\perp$*; if* δ *was the output of* anonSign *on input* (i, M) *it returns* pk_i *(the public-key of the user on whose behalf* anonSign *produced a signature), otherwise (if* δ *was not produced by* anonSign*) the oracle behaves as* Open *as long as* Open *returns some* pk_i *such that* $i \in Corr$ *(otherwise, if* $i \notin Corr$ *the oracle returns* $\perp$*). (2)* anonMining *is given as input a sequence of valid signatures* $\delta_1, \ldots, \delta_K$*; if* δ_j *was the output of* anonSign *on input* (i, M)*, set* $L_j = i$*. Otherwise, compute* $i = \mathsf{tableLook}(\mathsf{Open}(\delta_j))$ *and if* $i \in Corr$ *set* $L_j = i$*; if on the other hand,* $i \notin Corr$ *return* $\perp$*. Finally, the* anonMining *oracle returns* $\mathsf{usageChar}(L_1, \ldots, L_K)$.

A data mining group signature, satisfies anonymity if there exists an oracle anonSign *such that for the oracles* anonOpen, anonMining *as defined above it holds that any* PPT *anoymity adversary* $\mathcal{A}$ *cannot distinguish between these three oracles and the* OSign, Open, Mining *oracles.*

Some remarks about the definition above are in place: The definition of anonymity is in the sense of indistinguishability between the real implementation of Mining and Open functionalities and an idealized version of them. This allows maximum flexibility in designing distributed datamining schemes.

The definition of anonymity implies the non-malleability of the encryption algorithm employed in DMGS, since if an adversary is capable of modifying the signature of an uncorrupted user without affecting its validity, this would force the anonOPEN oracle to return $\perp$ (something that would not occur in the case of the OPEN oracle). Furthermore, observe that the definition of anonymity implies that the encryption algorithm satisfies IND-CPA security (as all signatures and hence ciphertexts can be simulated by anonSign without the plaintext information that corresponds to the signer's identity).

Note that the functions of anonOpen, anonMining can consult the correspondence between simulated signatures and identities and thus maintain correctness as in a real world execution.

4 Data Mining Group Signatures: Efficient Construction

Here we present a concrete DMGS scheme based on the short group signature of Boneh et al. [4], which is based on bilinear maps. The public parameters of the scheme are the following:

(p1) Two groups of order p where p is a ℓ_p-bit prime, denoted by $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$, so that $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map such that (1) for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$, and (2) $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$. Moreover, let ψ be a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(g_2) = g_1$.

(p2) An elliptic curve group of prime order q where q is ℓ_q-bit prime, denoted by $\langle G_1 \rangle$, over which the Decisional Diffie-Hellman is hard.

We assume $p = q$ for simplicity but our construction can also be ported to the more general setting that different size groups are being used. We notice that ℓ_p can be quite small, e.g., the order of 170 bits is sufficient. Now we specify the signature scheme.

$\mathsf{KeyG}_{\mathsf{DMGM}}$: The public parameters are selected as described above in (p1) and (p2). The key-generator selects $\gamma \leftarrow_R \mathbb{Z}_p$ and sets $w = g_2^{\gamma}$. The key $\mathcal{Y}_{\mathsf{DMGM}}$ is set to $\langle g_1, g_2, w, u, \mathrm{desc}(\mathbb{G}_1||\mathbb{G}_2||\mathbb{G}_T||e), n\rangle$, where $\mathrm{desc}(\cdot)$ is a description of the given groups including membership test and definition of group operation, and the users' secret keys are set to $\mathsf{sk}_i = \langle x_i, \sigma_i = g_1^{1/(\gamma+x_i)}\rangle$ for $x_i \leftarrow_R \mathbb{Z}_p$. Note that $e(\sigma_i, g_2^{x_i} w) = e(g_1, g_2)$ is the property satisfied by all user secret keys, and that σ_i or G^{x_i} will uniquely identify the user. We also let $u \in \mathbb{G}_1$ to be a generator of the group. The DMGM maintains a user database that contains entries of the form $\{(\mathsf{id}_i, G^{x_i})\}_i$. The algorithm $\mathsf{tableLook}(\cdot)$ on input G^{x_i} will return the identity id_i. Note that id_i, G^{x_i} can be required to be digitally signed by the user so that non-repudiation is facilitated (but we do not consider this aspect in our current modeling).

$\mathsf{KeyG}_{\mathsf{DM}}$: here we use the m-server protocols `ExpGen` and `PkGen2` from section 2 over the group $\langle G_1\rangle$. Recall that the protocol is based on parameters m, t and will produce the values $H_0, H_1, \ldots, H_{t-1} \in \langle G_1\rangle$ as well as the private output $\mathcal{S}_i$ for each server i such that $G^{\mathcal{S}_i} = H_0 H_1^i \ldots H_{t-1}^{i^{t-1}}$. The public-key that will be used for encryption will be $H_0 = G_1^s$ with $s \overset{(t,m)}{\longleftrightarrow} (\mathcal{S}_1, \ldots, \mathcal{S}_m)$. Additionally the protocol `PkGen2` produces value $G_2 \in \langle G_1\rangle$.

Remark 1. If the DMGM should not know the private keys of the users (i.e., if the property of exculpability is required), then according to [4] one can achieve this by extending the above $\mathsf{KeyG}_{\mathsf{DMGM}}$ algorithm as follows: instead of giving user i the private key $(\sigma_i = g_1^{1/(\gamma+x_i)}, x_i)$, the user and the key issuer can execute an interactive protocol so that at the end user i will obtain a triple (σ_i, x_i, y_i) such that $\sigma_i^{\gamma+x_i} h_1^{y_i} = g_1$, where $h_1 \in \mathbb{G}_1$ is a public parameter, and $y_i \leftarrow_R \mathbb{Z}_p$ is chosen by the user and kept secret from the group manager. If done so, the `SIGN` protocol below needs to be extended correspondingly (but this can be done in a straightforward manner).

Sign: Given a user's secret key $\langle x, \sigma\rangle$ and a message M. The signing algorithm will be obtained by applying the Fiat-Shamir heuristics on an appropriately selected proof of knowledge. The proof will also be helpful for the non-malleability aspects of the ciphertext that is embedded into the signature. Below we explain this proof in detail. First, the signer computes the following values: $T_1 = g_1^z u^{z'}$, $T_2 = g_1^{z'} \sigma$, $T_3 = G_1^r$, $T_4 = G_2^r$, $T_5 = H_0^r G^x$, where $r, z, z' \leftarrow_R \mathbb{Z}_p$.

Subsequently the signer will construct the signature on a given message M by providing a proof for a suitable set of relations; the signer knows the witnesses $v_x, v_z, v_{z'}, v_{xz}, v_{xz'}, v_r$ that satisfy the following relationships: $T_1 = g_1^{v_z} u^{v_{z'}}$ in $\mathbb{G}_1$, $T_3 = G_1^{v_r}$ in $\langle G_1\rangle$, $T_4 = G_2^{v_r}$ in $\langle G_1\rangle$, $T_5 = H_0^{v_r} G^{v_x}$ in $\langle G_1\rangle$, $T_1^{v_x} = g_1^{v_{xz}} u^{v_{xz'}}$ in $\mathbb{G}_1$, and $e(T_2, g_2)^{v_x} \cdot e(g_1, g_2)^{-v_{xz'}} \cdot e(g_1, w)^{-v_{z'}} = e(g_1, g_2)/e(T_2, w)$ in $\mathbb{G}_T$.

As a consequence, the signature is constructed as follows: first the values $\rho_z, \rho_{z'}, \rho_x, \rho_{xz}, \rho_{xz'} \leftarrow_R \mathbb{Z}_p$ and $\rho_r \leftarrow_R \mathbb{Z}_q$ are selected. Then the following values are computed:

$$R_1 = g_1^{\rho_z} u^{\rho_{z'}},\ R_2 = e(T_2, g_2)^{\rho_x} \cdot e(g_1, g_2)^{-\rho_{xz'}} \cdot e(g_1, w)^{-\rho_{z'}}$$

$$R_3 = G_1^{\rho_r},\ R_4 = G_2^{\rho_r},\ R_5 = H_0^{\rho_r} G^{\rho_x},\ R_6 = T_1^{\rho_x} g_1^{-\rho_{xz}} (u)^{-\rho_{xz'}},$$

Then we employ the hash function $\mathcal{H}'$ to compute

$$c \leftarrow \mathcal{H}'(M||\mathcal{Y}||M||T_1||T_2||T_3||T_4||T_5||R_1||R_2||R_3||R_4||R_5||R_6)$$

Subsequently the following values are computed: $s_x = \rho_x + cx$ in $\mathbb{Z}_p$, $s_z = \rho_z + cz$ in $\mathbb{Z}_p$, $s_{z'} = \rho_{z'} + cz'$ in $\mathbb{Z}_p$, $s_{xz} = \rho_{xz} + cxz$ in $\mathbb{Z}_p$, $s_{xz'} = \rho_{xz'} + cxz'$ in $\mathbb{Z}_p$, $s_r = \rho_r + cr$ in $\mathbb{Z}_q$. The output of the signing algorithm is the tuple: $\delta = \langle T_1, T_2, T_3, T_4, T_5, c, s_x, s_z, s_{z'}, s_{xz}, s_{xz'}, s_r \rangle$.

Verify: Signature verification is achieved by the following test: $c \stackrel{?}{=} \mathcal{H}'\Big(\mathcal{Y}||M||T_1||T_2||T_3||T_4||T_5||g_1^{s_z} u^{s_{z'}} T_1^{-c}||\tilde{E}||G_1^{s_r} T_3^{-c}||G_2^{s_r} T_4^{-c}||T_5^{-c} G^{s_x} H_0^{s_r}||T_1^{s_x} g_1^{-s_{xz}} u^{-s_{xz'}}\Big)$, where $\tilde{E} = e(T_2, g_2)^{s_x} \cdot e(g_1, g_2)^{-s_{xz'}} \cdot e(g_1, w)^{-s_{z'}} \cdot (e(g_1, g_2)/e(T_2, w))^{-c}$.

`OPEN`: this m-server protocol is a modification of the `DEC` protocol of section 2. The only essential difference is that the test for signature validity based on the public-key $\mathcal{Y}$ substitutes the verification of the proof that accompanies the ciphertext there. Observe that the tuple $\mathsf{mt} = (T_3, T_4, T_5)$ can be parsed out of δ and corresponds to a reduced ciphertext in the terminology of section 2. The mining servers execute the `DEC` protocol as described there to produce the value $G^{x_i} = \mathtt{DEC}(\mathsf{mt})$ that can be used to identify the signer in conjunction with the `tableLooK` function. Recall that the user database contains entries of the form $\{(\mathsf{id}_i, G^{x_i})\}_i$ and the table can be queried by the function $\mathsf{tableLook}(\cdot)$ that on input G^{x_i} will return id_i.

`MINING`: The `MINING` protocol will be an m-server protocol that is given as input a vector of signatures $(\delta_1, \ldots, \delta_K)$ (with the precondition that they are valid) out of which their mining tags $(\mathsf{mt}_{\delta_1}, \ldots, \mathsf{mt}_{\delta_K})$ can be parsed; these correspond to reduced ciphertexts of the underlying encryption scheme. We present two different `MINING` protocols that employ the `MIX`, `DEC`, `CMP` as sub-protocols operating over the mining tags that are parsed from the given vector of valid signatures.

(I) *Usage histogram:* The usage histogram functionality asks for a histogram $(\mathsf{id}_i, \mathsf{count}_i)$ for $i = 1, \ldots, n$ where count_i is the number of signatures signer id_i contributed.

Code snippet for `MINING`: usage histogram
Parse $\delta_1, \ldots, \delta_K$ to obtain $\mathsf{mt}_{\delta_1}, \ldots, \mathsf{mt}_{\delta_K}$;
$\langle \mathsf{mt}'_1 \ldots \mathsf{mt}'_K \rangle \leftarrow \mathtt{MIX}(\mathsf{mt}_{\delta_1}, \ldots, \mathsf{mt}_{\delta_K})$;
for $i = 1$ to K do $\mathsf{id}_i \leftarrow \mathsf{tableLook}(\mathtt{DEC}(\mathsf{mt}'_i))$;
$\mathtt{SORT}(\mathsf{id}_1, \ldots, \mathsf{id}_K)$;

The complexity of the usage-histogram functionality is equal to $\mathcal{O}(K \log K + K dec + mix(K))$, where dec is the cost of DEC and mix is the cost of MIX (note that SORT is implemented locally by each server on its local output).

(II) *Blinded usage histogram with outlier detection* : Given the sequence of signatures, we want to create a histogram that contains entries of the form (i, count_i) where i corresponds to one of the signers that produced some of the signatures among $\delta_1, \ldots, \delta_K$ and count_i corresponds to the number of signatures that were contributed by this user; note that here i does not identify the user (it is not equal to the user's identity and not correlated with it — it is simply a histogram-specific pseudonym for the user and is used only for the presentation of the histogram). Besides this, detecting outliers requires the mining servers to discover the identity of signers that either over-use (e.g. spammers) or under-use (e.g. those that haven't seen enough advertisements) the system. To facilitate this the algorithm takes two parameters $\mathsf{lo}, \mathsf{hi} \in \{1, \ldots, K\}$ and requires the recovery of the identity of any signer whose usage is below (resp. above) lo (resp. hi):

Code snippet for MINING blinded usage histogram with outlier detection

```
Parse δ_1, ..., δ_K to obtain mt_{δ_1}, ..., mt_{δ_K};
⟨mt'_1 ... mt'_K⟩ ← MIX(mt_{δ_1}, ..., mt_{δ_K});
for i = 1 to K do
        k = 0;
        if mt'_i ≠ NIL then
                k = k + 1;
                count[k] = 1;
                for j = i + 1 to K do
                        if mt'_j ≠ NIL then
                                test ← CMP(mt'_i, mt'_j);
                                if test == 1 then
                                        count[k] = count[k] + 1;
                                        mt'_j = NIL;
                output ⟨k, count[k]⟩;
                if (count[k] < lo) or (count[k] > hi)
                    then output id ← tableLook(DEC(mt'_i));
```

The complexity of the blinded usage histogram generation is $\mathcal{O}(K^2 cmp + mix(K))$, where cmp is the cost of CMP.

Correctness and Security Arguments. Below we argue how our construction specified above satisfies the model that we put forth in sections 2 and 3.

Our security arguments will be split into steps. First we will show that our suite of multi-server protocols is t-distribution safe for $m \geq 2t - 1$ servers. This will enable us to reduce any adversary that takes advantage of the multi-server nature of the system and corrupt a set of $t - 1$ servers to an adversary that performs the same attack against a single honest server. Then, we will show that our scheme in the single server setting satisfies the properties we put forth in section 3.

Theorem 2. *The suite of m-server protocol suite* $\langle \mathtt{KeyG}_{\mathsf{DM}}, \mathtt{OPEN}, \mathtt{DEC}, \mathtt{MIX}, \mathtt{CMP} \rangle$ *is t-distribution-safe under the discrete-logarithm assumption and that* $\mathcal{H}$ *is a random oracle that is controlled by the simulator provided that* $m \geq 2t - 1$.

Theorem 3. *The DMGS scheme introduced above satisfies (i) correctness, (ii) traceability, (iii) anonymity, based on: the Strong-Diffie Hellman Assumption over* $\mathbb{G}_1, \mathbb{G}_2$*, the Decisional Diffie-Hellman Assumption over* $\langle G_1 \rangle$ *and the assumption that* $\mathcal{H}'$ *is modeled as a random oracle.*

5 Conclusion

We conceptualized the notion of privacy preserving data mining within anonymous credential systems. We advocated this general notion as fundamental to adapting anonymous credentials in general systems. We then instantiated it in the context of group signatures as "data mining group signatures" and presented two particular instantiations of it. We included a modeling of the notion, introduced distribution safety as a way to modularly argue the security of distributed cryptosystems and we presented an explicit construction of our notion. A number of issues remain for further investigation: Extending the mining instances to other cases crucial in transaction systems while maintaining efficiency is an important direction. Our definition of the notion is based on combining the properties of correctness, traceability and anonymity; considering attackers that adaptively decide which of the properties to violate (as in a simulation-based "ideal functionality" formulation) is an open subject to consider. Even further, presenting constructions without the random oracle idealization or in a fully concurrent execution model are open questions as well.

References

1. Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the k th-ranked element. In: Cachin and Camenisch [5], pp.40–55
2. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880. Springer, Heidelberg (2000)
3. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003, Warsaw, Poland. LNCS, vol. 2656. Springer, Heidelberg (2003)
4. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
5. Cachin, C., Camenisch, J. (eds.): Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, nterlaken, Switzerland, May 2-6, 2004. LNCS, vol. 3027. Springer, Heidelberg (2004)
6. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clonewars: efficient periodic n-times anonymous authentication. In: Juels, A., Wright, R.N., di Vimercati, S.D.C. (eds.) ACM Conference on Computer and Communications Security, pp. 201–210. ACM, New York (2006)

7. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact e-cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
8. Chaum, D.: Blind signatures for untraceable payments. In: Crypto (1982)
9. Chaum, D.: Security without identification: Transactions systems to make big brother obsolete. C. ACM 28(10), 1030–1044 (1985)
10. Chaum, D.: Showing credentials without identification. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 241–244. Springer, Heidelberg (1986)
11. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403. Springer, Heidelberg (1990)
12. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
13. Damgård, I., Dupont, K., Pedersen, M.Ø.: Unclonable group identification. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 555–572. Springer, Heidelberg (2006)
14. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: Proceedings of the 28th Symposium on Foundations of Computer Science (FOCS), pp. 427–437. IEEE Computer Society Press, Los Alamitos (1987)
15. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
16. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin and Camenisch [5], pp.1–19
17. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
18. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (1999)
19. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y. (ed.) Public Key Cryptography. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2003)
20. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Grossman, R., Bayardo, R., Bennett, K.P. (eds.) KDD, pp. 593–599. ACM, New York (2005)
21. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 571–589. Springer, Heidelberg (2004)
22. Kissner, L., Song, D.X.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
23. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880. Springer, Heidelberg (2000)
24. Lindell, Y., Pinkas, B.: Privacy preserving data mining. J. Cryptology 15(3), 177–206 (2002)
25. Liu, X., Yang, X., Wetherall, D., Anderson, T.: Efficient and secure source authentication with packet passports. In: Proceedings of 2nd USENIX Steps to Reduce Unwanted Traffic on the Internet workshop (SRUTI 2006) (2006)
26. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Samarati, P. (ed.) Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, PA, USA, November 2001, pp. 116–125. ACM Press, New York (2001)

27. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract). In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
28. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
29. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptology 15(2), 75–96 (2002)
30. Stadler, M., Piveteau, J.-M., Camenisch, J.: Fair blind signatures. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, Springer, Heidelberg (1995)
31. Yang, X., Wetherall, D., Anderson, T.: A dos-limiting network architecture. In: ACM SIGCOMM, pp. 241–252 (2005)
32. Yang, Z., Zhong, S., Wright, R.N.: Privacy-preserving classification of customer data without loss of accuracy. In: SIAM International Data Mining Conference (2005)

Improved Privacy of the Tree-Based Hash Protocols Using Physically Unclonable Function*

Julien Bringer[1], Hervé Chabanne[1], and Thomas Icart[1,2]

[1]Sagem Sécurité
[2]Université du Luxembourg
firstname.name@sagem.com

Abstract. In 2004, Molnar and Wagner introduced a very appealing protocol dedicated to the identification of RFID tags. Their scheme relies on a binary tree of secrets which are shared – for all nodes except the leaves – amongst the tags. Hence the compromise of one tag also has implications on the other tags with whom it shares keys. We describe a new man-in-the-middle attack against this protocol which allows to break privacy even without opening tags. Moreover, it can be applied to some other RFID protocols which use correlated keys as the one described recently by Damgård and Pedersen at CT-RSA 2008.

We introduce a modification of the initial scheme to allow us to thwart this and to strengthen RFID tags by implementing secrets with Physical Obfuscated Keys (POKs). This doing, we augment tags and scheme privacy, particularly general resistance against physical threats.

Keywords: RFID tags, Tree-Based Hash Protocol, POK, PUF, Privacy.

1 Introduction

Radio Frequency Identification (RFID) tags are made of a small chip containing a unique identification number. They communicate in the air with the system via a reader. One of their main applications is to track objects on which they are attached.

RFID systems have to deal with the scarcity of tags resources as well as the privacy needed for tag identification. In [10,11], a protocol which seems well suited to handle these two constraints has been introduced. Indeed, the identification protocol of Molnar *et al.* requires only limited cryptographic functionality and has some useful properties such as the delegation of some identifications from a Trusted Center to readers. This protocol relies on a binary tree of secrets. The secret corresponding to a leaf is uniquely associated to one tag, but all the other secrets in the tree are shared with different tags. Thus, as it is studied in [4,12,13], the compromise of the keying material of some tag leads to learn the shared keys with some other tags. If many tags are compromised, this could allow to track some non-compromised tags. This can be considered as a main

* This work was partially supported by the french ANR RNRT project T2TIT.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 77–91, 2008.

threat to the privacy of the system. This problem has already been addressed in [2], the compromise of tags still leaks information about the keying material of the system.

To thwart this, we want to increase the resistance of tags against physical threats. Physical Obfuscated Keys (POKs) have been introduced by Gassend [6] as a mean to securely store a secret inside a chip. They are strongly related to Physical Unclonable Functions (PUFs). Indeed, POKs were introduced as a proposition to implement keys in a more secure manner. They are built such that their observations by an adversary corrupt the chip and then destroy them. Note that the use of PUFs inside RFID tags has already been considered in [3,15].

The main achievement of this paper is to describe how to replace each secret by two POKs during the Tree-Based Hash protocol. They are activated alternately and each one taken separately does not reveal anything on the secret. Cryptographic computations are carried out with two steps, where during a step, only one POK is activated. Moreover an adversary can gain access only to one POK by sacrificing the chip. By construction the underlying key is thus safe from this compromise of one POK.

Our paper is as follows. In Sect. 2, we recall the principles of the Tree-Based Hash protocol [11] and those of POKs. In Sect. 3 and 4, we describe our privacy model. In Sect. 5, we explain why some private informations leak with a Tree-Based Hash protocol. In fact, we show a new attack against [11] and [5] where an adversary is able to track tags even without compromising any tags. In Sect. 6, we describe our modification of the protocol. Section 7 examines the security of our proposition and Section 8 examines the privacy of our scheme to formally prove the latter in the random oracle model. Section 9 concludes. Security proofs and practical implementations are sketched in appendices A and B.

2 Preliminaries

2.1 The Protocol [11] in a Nutshell

In the following, we describe the general principles of the Tree-Based Hash protocol and invite the readers to go through [11] to get full details.

During system initialization, a Trusted Center generates a tree of secrets (keys), for instance a binary one. Each leaf is associated to a tag. A tag knows all keys $K_1, \ldots, K_d$ along the path from the root to its leaf. Let F denotes an appropriate public pseudo-random function. When a tag is challenged by a reader which sends to it a random value r, it responds by generating a new value each time – $F_{K_1}(r, r')$, $F_{K_2}(r, r')$, ..., $F_{K_d}(r, r')$ – where r' is another random value generated and transmitted by the tag. The Trusted Center can easily check to which key corresponds the received value in its tree of secrets by verifying for a given (r, r'):

1. to which node corresponds $F_{K_1}(r, r')$,
2. between the 2 children of this node, which one is associated with $F_{K_2}(r, r')$,
3. repeat this verification, level after level from the root to the leaves,
4. and then identify which leaf (tag) comes with $F_{K_d}(r, r')$.

A Practical Example. To get a better idea of the involved figures, we take back the example given in [11]. They have 2^{20} tags. The binary tree is replaced by a tree with a branching factor $Q = 2^{10}$ and is made of two levels. Each tag stores two 64-bit secrets. Using a Tree-Based identification protocol enables to reduce the number of tests a Trusted Center needs to do. In this example, a Trusted Center has to compute only 2×2^{10} times the function F, 2^{10} for each round, instead of 2^{20} without this protocol. This improvement is very interesting, because if the system's size is S, the number of computation for the Trusted Center is always in $O(log_Q(S)Q)$ computation.

It should be noted that this protocol is very similar to a popular RFIDs singulation algorithm: the tree walking algorithm [1]. Using this protocol leads to an optimized singulation.

2.2 Physical Unclonable Function and Physically Obfuscated Key

Gassend in [6] introduces the concept of PUF. A Physical Unclonable Function (PUF) is a function that maps challenges (stimuli) to responses, that is embodied by a physical device, and that has the following properties:

1. easy to evaluate,
2. hard to characterize, from physical observation or from chosen challenge-response pairs,
3. hard to reproduce.

For a given challenge, a PUF always gives the same answer. The hardness of characterization and the reproduction is hard; i.e. it is impossible to reproduce or to characterize the PUF thanks to a reasonable amount of resources (time, money, ...). PUF can thus be viewed as pseudo-random function[1] where the randomness is insured thanks to physical properties. In the rest of this paper, PUFs are formalized as perfect random functions, i.e. functions with maximal output's entropy.

We also write $\mathsf{Gen}^{\mathsf{PUF}}(1^{2^k})$ for a generator of random, independent PUFs.

One kind of PUF, as mentioned in [15] as I-PUF for Integrated Physical Unclonable Function, has other interesting properties:

1. The I-PUF is inseparably bound to a chip. This means that any attempt to remove the PUF from the chip leads to the destruction of the PUF and of the chip.
2. It is impossible to tamper with the communication (measurement data) between the chip and the PUF.
3. The output of the PUF is inaccessible to an attacker.

These properties insure the impossibility to analyze physically a PUF without changing its output. Hence, physical attacks corrupt the PUF and the chip leaving the attacker without any information about the PUF. Particularly, volatile

[1] Note however that they can be limited in the number of possible challenge-response pairs as explained in [8].

memory cannot be read out without destroying the I-PUF. Silicon PUF have been already described in [7] and can be taken as relevant examples of I-PUF, they are based on delay comparison among signals running through random wires. Moreover, they only require a few resources to be implemented. A practical example of implementation is described in [14].

In [6], it is shown how to implement a key with a PUF, this implementation is called a Physically Obfuscated Key (POK), by applying a fixed hard-wired challenge to the PUF. In fact, using different challenges, several POKs can be obtained from one PUF. In the sequel, we refer to a POK as a value, stored in a tag, which is accessible only when the underlying PUF is stimulated.

2.3 How We Use POKs

The key has to be stored digitally when involved in some computations, whatever the use of the tag is. Consequently, it could be possible to get a dump of the volatile memory and then to obtain the value of the key. This type of attack has been considered in [2] with a general line of defense for POKs: split the computations with the key in two steps. Of course, the difficulty we encounter is to cope with cryptographic computations and to find a way to split them.

A key K of the tree would be hard-wired thanks to two POKs K' and K'' such that $K = K' \oplus K''$where the two parts K' and K'' are different for each tag.

Note that challenges used to stimulate the PUF to generate keys are stored in the tag. Because the equality $K = K' \oplus K''$ stands for all tags in the same branch, neither K' and K'' need to be known from the outside, nor pairs of input/output from the PUF do.

3 Security Model

Here we propose to apply to RFID systems the following security model for completeness and soundness. This is a simplification of [16].

3.1 Adversary Model

We sketch the possible actions of an adversary over a system. The system contains a Trusted Center TC which wants to communicate with N tags. We assume that the protocol is a challenge-response protocol: to authenticate a tag, the Trusted Center sends a challenge and then waits for a response from the tag.

- SendTC: this function enables the adversary to interact with the TC. Using this function, he gets a challenge a_0 and he possibly tries to answer by playing the role of a tag, in order to gain information over the key material. Nevertheless, he does not receive the result of the identification.
- SendTag: this function enables an adversary to communicate with a tag. SendTag$(\mathcal{T}, a_0)$ means the adversary sends a_0 to the tag $\mathcal{T}$. This leads to the complete output from the tag.

- RESULT: this function allows an adversary to determine whether a bit string, taken as input by the function, is a valid communication transcript of the protocol. RESULT gives the authentication result the TC would have produced for a sent challenge and a response from a tag which are read in the input bit string.
- CORRUPT: this function enables the adversary to open a tag to get all the memory, volatile and non-volatile. CORRUPT enables an adversary to get keys – if any – inside a tag and to get the volatile memory at any moment of the tag computation.

We also suppose that an adversary has access to any random oracle which may be used in the protocol.

3.2 Completeness

Definition 1. *The scheme is* **complete** *when the probability of a genuine tag to fail during the identification process is negligible. I.e. for all tags* $\mathcal{T}$*,*

$$\Pr\left(\textsc{Result}\left(a_0^{\mathsf{TC}}, \textsc{SendTag}\left(\mathcal{T}, a_0^{\mathsf{TC}}\right)\right) = false \,|\, a_0^{\mathsf{TC}} = \textsc{SendTC}\left(\right)\right)$$

is negligible.

3.3 Soundness

Definition 2. *The scheme is* **sound***, if any polynomially bounded adversary* $\mathcal{A}$ *cannot produce a valid communication transcript* $\mathcal{C}_{\mathcal{A}}$*, except with a negligible probability. Furthermore,* $\mathcal{C}_{\mathcal{A}}$ *must neither lead to the identification of a corrupted tag nor be an eavesdropped communication. I.e.*

$$\Pr\left(\textsc{Result}\left(\mathcal{C}_{\mathcal{A}}\right) = true\right)$$

is negligible.

These definitions are the adaptation of the usual correctness and soundness in the model. Correctness ensures a legitimate tag identifies itself with an overwhelming probability. Soundness ensures that no adversary can impersonate a tag. Nevertheless, in the definition of soundness, we assume that adversaries are active. For instance, they can impersonate a TC or eavesdropped communications or even corrupt tags to get information on secrets of the system.

4 Privacy Model

We present here our model of privacy. To define privacy, we define a game. An adversary relevant against privacy is able to win this game with a non negligible probability.

Thanks to the experiment described in Fig. 1, $\mathcal{A}$ is an adversary who wants to find a privacy leakage in the protocol (where $\overset{\mathcal{R}}{\leftarrow}$ denotes an element taken at

random). The privacy is defined as the advantage of the adversary over two tags amongst two systems of tags he had chosen. If the advantage of $\mathcal{A}$ is negligible, this means he is not able to link any tag inside S_1 and S_2. If $\mathcal{A}$ is relevant for this game, he is able to construct subsystems with a special property: given a tag, he can determine in which subsystem it belongs. This definition is more general than anonymity and untraceability. If tags can be identified from an adversary or can be traced, it is easy for an adversary to construct subsystems in order to succeed at our game.

Experiment $\mathsf{Exp}^{priv}_{\mathcal{A},S}$:

Setup:

1. Initialize one system S.

Phase 1 (learning):

1. $\mathcal{A}$ may do the following in any interleaved order:
   (a) make arbitrary SendTag queries to any tag in S,
   (b) make arbitrary SendTC queries,
   (c) make arbitrary Result queries,
   (d) make arbitrary Corrupt queries to any tag in S,
   (e) make arbitrary calls to the random oracle.

Phase 2 (challenge):

1. $\mathcal{A}$ selects two subset of S: S_1 and S_2,
2. $\mathcal{A}$ selects two non corrupted tags $\mathcal{T}_1 \in S_1$ and $\mathcal{T}_2 \in S_2$.
3. Remove $\mathcal{T}_1$ and $\mathcal{T}_2$ from S_1 and S_2.
4. Let $b \overset{\mathcal{R}}{\leftarrow} \{1, 2\}$ to select $\mathcal{T}_b$ one of these tags.
5. $\mathcal{A}$ may do the following in any interleaved order:
   (a) make arbitrary SendTag queries to any tag in $S_1 \backslash \mathcal{T}_1$, $S_2 \backslash \mathcal{T}_2$ and $\mathcal{T}_b$,
   (b) make arbitrary SendTC queries,
   (c) make arbitrary Result queries,
   (d) make arbitrary Corrupt queries to any tag in $S_1 \backslash \mathcal{T}_1$, $S_2 \backslash \mathcal{T}_2$,
   (e) make arbitrary calls to the random oracle.
6. $\mathcal{A}$ outputs a guess index b'

$\mathsf{Exp}^{priv}_{\mathcal{A},S}$ succeeds if $b = b'$.

Fig. 1. Privacy Experiment

Definition 3. *A protocol in a RFID system is* **private** *if for a polynomially bounded adversary $\mathcal{A}$ following the experiment $\mathsf{Exp}^{priv}_{\mathcal{A},S}$, then*

$$|\Pr[b' = b] - \Pr[b' \neq b]|$$

is negligible.

In each step, $\mathcal{A}$ is allowed to use the random oracle, but we omit it to simplify.

This privacy definition is more general than the privacy definition of Juels and Weis in [9]. This is a consequence of the possibility to consider shared keys inside tags whereas it is not taken in account in their model. Indeed in their model, they suppose keys inside tags are all independent. In this case, it is unnecessary

to consider the whole system to determine whether an adversary has advantages on distinguishing two tags, whereas it is an important threat to consider in Tree-Based protocols. Furthermore, the original Tree-Based Hash protocol is private in their model although it is not in ours (cf. Sect. 5).

Vaudenay in [16] defines a new model of privacy. Privacy is defined as a leakage of information of the whole system. In the Vaudenay's model, a system of tags is private if it is possible to perfectly simulate the system. An adversary should not be able to distinguish whether he is attacking a legitimate system or a simulated one. From now on, this seems to be the most general model as it is clear that a privacy leakage is a gain of information on the system. Nevertheless, a system could not be perfectly simulated – as it is the case for our scheme introduced in Sect. 6 when we allow the adversary to use the RESULT oracle – without implying that there exists a way to obtain information over tags inside the system. That is why we introduce our privacy definition which can be seen as a kind of trade-off between [9] and [16].

5 A New Privacy Leakage Against Tree-Based Hash Protocols

The original Tree-Based Hash protocol proposed in [11] had been proved to have some privacy leakage in [4,12,13]. Opening a tag, while keys are not protected, leads to the knowledge of shared keys in the system. Note that in one version of the protocol in [11](the one described in section 2.1), there are cases where it is possible to determine whether two tags share keys even without getting physically the keys.

Let us denote $C_1^{\mathcal{T}}(r) = F_{K_1}(r, r'), \ldots, C_d^{\mathcal{T}}(r) = F_{K_d}(r, r')$ the outputs of the tag $\mathcal{T}$ for the challenge r. $C_i^{\mathcal{T}}(r) = F_{K_i}(r, r')$ is the output needed to authenticate at the depth i in the tree of keys. Suppose the $C_i^{\mathcal{T}}(r)$ are independent from each other. As a consequence, $C_i^{\mathcal{T}}(r)$ can be computed without the knowledge of $C_1^{\mathcal{T}}(r), \ldots, C_{i-1}^{\mathcal{T}}(r), C_{i+1}^{\mathcal{T}}(r), \ldots, C_d^{\mathcal{T}}(r)$. In this case, using one RESULT query and two SENDTAG queries, it is possible to determine whether two tags share one key.

Getting a random challenge r from SENDTC, the adversary can use SENDTAG$(\mathcal{T}, r)$ and SENDTAG$(\mathcal{T}', r)$. He is then in possession of two communications $C_1^{\mathcal{T}}(r), \ldots, C_d^{\mathcal{T}}(r)$ and $C_1^{\mathcal{T}'}(r), \ldots, C_d^{\mathcal{T}'}(r)$. For instance, to test whether the two tags have the same first key, the adversary uses the RESULT query on the communication $\left(r, C_1^{\mathcal{T}'}(r), C_2^{\mathcal{T}}(r), \ldots, C_d^{\mathcal{T}}(r)\right)$. If this communication is an admissible one, this means $\mathcal{T}$ and $\mathcal{T}'$ share the same first key. Otherwise, they do not. Of course, it is possible to do the same for a key at a different position.

This attack is practically feasible as the adversary only needs to interact with two tags and a reader. In fact, it is a general privacy threat that concerns RFID systems using correlated keys inside tags. As soon as the different components of a response (the C_i above) are not linked together, an adversary can mix the answers of several tags to learn if they share keys.

For instance it is the case of the new protocol recently introduced in [5]: It is a protocol with correlated keys, but unlike [11] it does not rely on a tree of secrets

in order to increase the possible choices of tuples of keys associated to tags, which allows to increase the resistance against corruption. However messages answered by a tag are still independent and the technique above still attacks the privacy of the scheme. In the next section, our protocol is constructed to avoid also this kind of vulnerability.

6 Our Proposition

6.1 System Parameters

Because of PUF and use of different random values for each key inside a tag, our protocol strengthens tags against the privacy leakage described in the previous section (see section 8 for this result).

We now give the parameters of our scheme. Our RFID system is made of N tags, the tree of key has a branching factor Q and a depth d. We use a pseudo-random function H implemented by a hash function.

The length of the random challenge a_0 sent by the TC is l_r, the length of keys is l_K and the length of the output of the hash function is l_H. The number of tags N is usually smaller than 2^{40}. A probability is considered negligible as soon as it is negligible in at least one of the following parameters: N, l_r, l_K, l_H. These systems is denoted $S(N, Q, d, l_r, l_H, l_K)$.

Setup. To create our system of tags, we need a generator function: $\mathsf{Gen}(1^k)$ outputs a random element of size k. To create our system of tags, we first use $Q + Q^2 + \ldots + Q^d$ times the function Gen to create our tree of keys. Each key is an output of $\mathsf{Gen}(1^{l_K})$. During the creation of a new tag, a set of keys is given, which enabled it to identify itself. The set of keys is made thanks to our tree of keys, which means it represents a path from the root to a leaf. All the tags have of course different sets of keys, with possibly $d-1$ keys shared. For one tag $\mathcal{T}$, it is denoted as $K_1^{\mathcal{T}}, \ldots, K_d^{\mathcal{T}}$. A tag is created with a new PUF obtained from $\mathsf{Gen}^{\mathsf{PUF}}(1^{2^{l_K}})$. As shown before, each key is implemented inside a tag via two POKs. To generate the value of these POKs, we once more use Gen. For each $K_i^{\mathcal{T}}$, $\mathsf{Gen}(1^{l_c})$ outputs a challenge c. This challenge is hard-wired with the PUF and outputs $\mathsf{PUF}(c) = K_i'^{\mathcal{T}}$. The couple of POKs associated to the key $K_i^{\mathcal{T}}$ is $(K_i'^{\mathcal{T}}, K_i''^{\mathcal{T}} = K_i'^{\mathcal{T}} \oplus K_i^{\mathcal{T}})$. As the PUF is considered to be a perfect random-function, $K_i'^{\mathcal{T}}$ and $K_i''^{\mathcal{T}}$ are considered to be random values of entropy l_K. This means that the knowledge of only one of these values does not reveal anything on $K_i^{\mathcal{T}}$.

6.2 The Protocol

In Fig. 2 is the description of our new protocol, where $\hat{K}_i^j$ denotes the key at the depth i on the j^{th} branch of the tree. The TC sends to the tag a challenge a_0, which is a random value. The tag $\mathcal{T}$ computes a random value $r_1^{\mathcal{T}}$ and sends $a_1 = H(a_0, r_1^{\mathcal{T}})$. The tag switches on the first POK to get $K_1'^{\mathcal{T}}$ and computes $A' = r_1^{\mathcal{T}} \oplus K_1'^{\mathcal{T}}$. This operation erases in volatile memory $r_1^{\mathcal{T}}$. The second POK is switched on to get $K_1''^{\mathcal{T}}$, and this erases $K_1'^{\mathcal{T}}$. Finally the tag computes $A'' =$

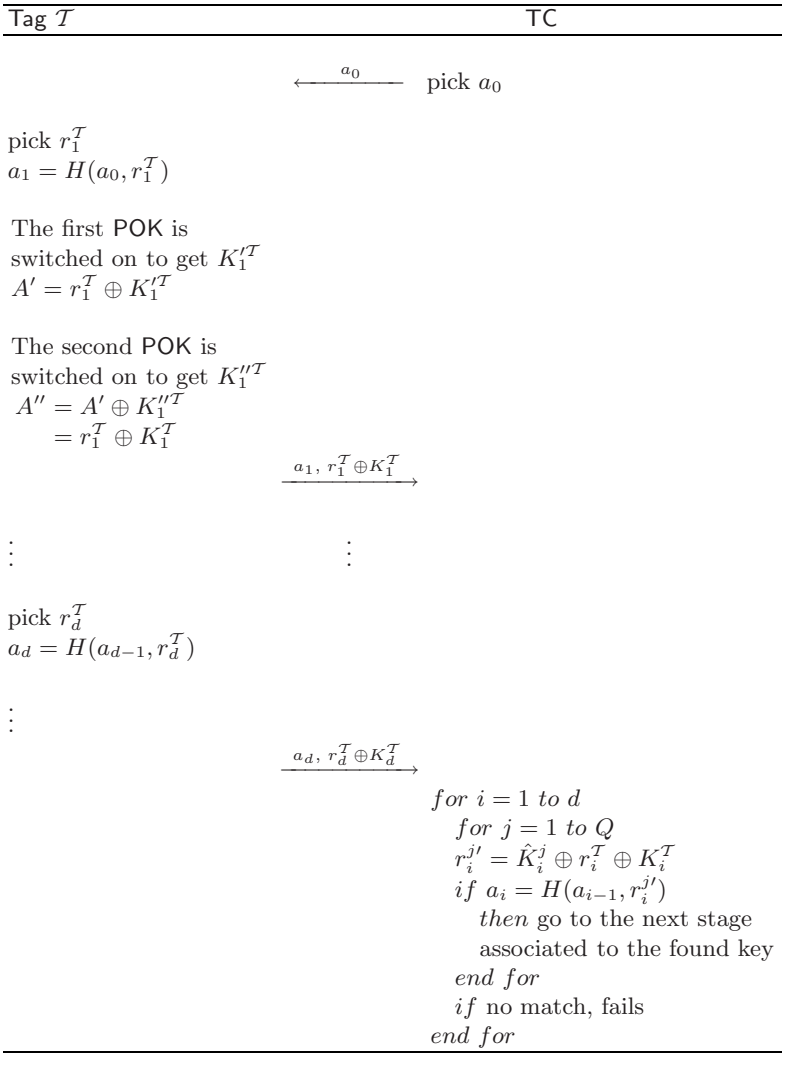


Fig. 2. The Identification Protocol

$A' \oplus K_1''^{\mathcal{T}}$ and sends $r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}}$. The tag picks a random value $r_2^{\mathcal{T}}$ and computes $a_2 = H(a_1, r_2^{\mathcal{T}})$ and sends it to the TC. It computes $r_2^{\mathcal{T}} \oplus K_2^{\mathcal{T}}$ using the same tricks as before. It repeats these operations $d-1$ times.

Then the Trusted Center (TC) tries amongst all the key $\hat{K}_1^j$ in the tree's first level whether it gets the equality $H(a_0, r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}} \oplus \hat{K}_1^j) = a_1$. If it finds one correct key, it searches the next key amongst the possible keys in the tree. If the operation is successful for the d levels then the tag is authenticated.

This protocol has all the advantages of the Tree-Based Hash protocols: it allows delegation and to have less computation for the TC than the exhaustive search but with an increased time of computation for the tag.

7 Security Analysis

In our protocol, SendTag outputs $(a_1, r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}}, \ldots, a_d, r_d^{\mathcal{T}} \oplus K_d^{\mathcal{T}})$ and Result returns whether the $2d+1$-tuple $(a_0, a_1, r_1^{\mathcal{T}} \oplus K_1^{\mathcal{T}} \ldots, a_d, r_d^{\mathcal{T}} \oplus K_d^{\mathcal{T}})$ is correct.

7.1 Restriction on the Corrupt Query Due to POKs

In our case, we make the hypothesis that corrupting a tag $\mathcal{T}$ leads an adversary to the knowledge of only one of the three possible type of sets:

1. a_{i-1}, $r_i^{\mathcal{T}}$ and $a_i = H(a_{i-1}, r_i^{\mathcal{T}})$,
2. a_{i-1}, $r_i^{\mathcal{T}}$, $a_i = H(a_{i-1}, r_i^{\mathcal{T}})$ and $r_i^{\mathcal{T}} \oplus K_i'^{\mathcal{T}}$,
3. a_{i-1}, $a_i = H(a_{i-1}, r_i^{\mathcal{T}})$, $r_i^{\mathcal{T}} \oplus K_i''^{\mathcal{T}}$ and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$.

Note that in the two first cases, an adversary does not learn the final output. Thanks to the possible actions of the adversary as defined in Sect. 3.1, we can prove:

Theorem 1. Corrupt *queries leak at most as many information on the key material as* SendTag *queries.*

Proof. Getting a_{i-1}, $r_i^{\mathcal{T}}$ and a_i is trivially of no interest. Getting a_{i-1}, $r_i^{\mathcal{T}}$, a_i and $r_i^{\mathcal{T}} \oplus K_i'^{\mathcal{T}}$ is equivalent as getting a_{i-1}, $r_i^{\mathcal{T}}$ and $K_i'^{\mathcal{T}}$. As $K_i'^{\mathcal{T}}$ is a random value of maximal entropy l_K, this leaks no information about $K_i^{\mathcal{T}}$. Finally, getting a_{i-1}, a_i, $r_i^{\mathcal{T}} \oplus K_i''^{\mathcal{T}}$ and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$ is equivalent as getting a_{i-1}, a_i, $K_i''^{\mathcal{T}}$ and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$. Because $K_i''^{\mathcal{T}}$ is a random value of entropy l_K, this is equivalent as getting a_{i-1}, a_i, and $r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$ which is exactly a part of an output of a SendTag query. □

In the sequel, we do not distinguish Corrupt from SendTag in proofs.

Remark 1. Formalizing Corrupt this way is convenient for our model and our proofs. The reality behind this formalization is still an open implementation issue. More concretely, the ability of an adversary to obtain a key from a POK without destroying the tag has to be evaluated precisely. This topic is however outside the scope of this paper.

7.2 Completeness and Soundness

Theorem 2. *Our scheme is complete. If H is preimage and collision resistant, then our scheme is sound.*

The proofs are available in Appendix A.

8 Privacy Analysis

As shown before in Sect. 5, this is an important point to determine whether an adversary gains any advantage using different outputs from different tags while he is using Result queries. In this paper, the protocol described has the property that an adversary cannot use different outputs from tags to make a new one which has a good probability of being admissible. This is shown in the following.

Proposition 1. *If H is collision resistant, an adversary, by mixing different outputs from different tags in a* Result *query gets a positive answer only with a negligible probability.*

Proof. $\mathcal{A}$ uses the SendTag query on tags in S. Then he uses the Result query. His query is of the form $a_0, \ldots, a_{i-1}^{\mathcal{T}^1}, r_{i-1}^{\mathcal{T}^2} \oplus K_{i-1}^{\mathcal{T}^2}, a_i^{\mathcal{T}^3}, r_i^{\mathcal{T}^4} \oplus K_i^{\mathcal{T}^4}, \ldots$ To be a valid communication, it has to exist a key $\hat{K}$ such that $a_i^{\mathcal{T}^3} = H(a_{i-1}^{\mathcal{T}^1}, r_i^{\mathcal{T}^4} \oplus K_i^{\mathcal{T}^4} \oplus \hat{K})$. We also have the equality $a_i^{\mathcal{T}^3} = H(a_{i-1}^{\mathcal{T}^3}, r_i^{\mathcal{T}^3})$. If the first equality occurs, while $\mathcal{T}^1$, $\mathcal{T}^3$ and $\mathcal{T}^4$ represent different tags, this leads to a collision on the output of H as $r_i^{\mathcal{T}^3}$ and $r_i^{\mathcal{T}^4}$ are generated randomly and $a_{i-1}^{\mathcal{T}^1}$ and $a_{i-1}^{\mathcal{T}^3}$ are outputs from H. So this proves this communication is valid with a negligible probability. □

Hence, in the sequel, we suppose an adversary never uses different SendTag outputs in one Result query.

Remark 2. Furthermore, if $\mathcal{A}$ tries some Result queries on a randomly modified communication from one tag, he gets a positive answer with a negligible probability. Consequently, Result query can just be used to verify whether a communication from **one** tag is valid or not.

Below, the random oracle H represents the hash function used in our protocol and we assume that the random generator in each tag is perfect.

Theorem 3. *Our protocol, in the random oracle model, is private.*

Proof. Let $L_{\mathcal{C}}^{i,lp}$ be the list of all the communications of $\mathcal{A}$ with tags in S_i during the learning phase and $L_{\mathcal{C}}^{i,cp}$ during the challenge phase except $\mathcal{T}_b$. Let $L_{\mathcal{C}}^{b}$ be the communication with $\mathcal{T}_b$. Let L_{R}^{lp} and L_{R}^{cp} the Result (or SendTC or Corrupt) queries used in the experiment. Let $L^{lp} = L_{\mathcal{C}}^{1,lp} \cup L_{\mathcal{C}}^{2,lp} \cup L_{\mathrm{R}}^{lp}$ and $L^{cp} = L_{\mathcal{C}}^{1,cp} \cup L_{\mathcal{C}}^{2,cp} \cup L_{\mathrm{R}}^{cp}$. Let L^1 be $L_{\mathcal{C}}^{1,lp} \cup L_{\mathcal{C}}^{1,cp}$ and L^2 be $L_{\mathcal{C}}^{2,lp} \cup L_{\mathcal{C}}^{2,cp}$.

To determine whether $\mathcal{T}_b$ is in S_1 or S_2, $\mathcal{A}$ has to determine whether $\mathcal{T}_b$ shares keys with tags in S_1 or in S_2. To achieve this, either he made some queries to the random oracle or not.

- **Case 1:** $\mathcal{A}$ did not make any random oracle query. So $\mathcal{A}$ can obtain a clue that $\mathcal{T}_b$ is in S_i just by looking at L^{lp}, L^{cp} and $L_{\mathcal{C}}^{b}$. We already proved in Sect. 8 that use of Result only helps to verify whether a communication from one tag is valid or not. To get a useful information, $\mathcal{A}$ has to compare the communications in L^1, L^2 and $L_{\mathcal{C}}^{b}$.

  To this aim, amongst triplets $(a_{i-1}^{\mathcal{T}}, a_i^{\mathcal{T}}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}})$, $\mathcal{A}$ needs to distinguish values which are correlated to the same keys. However, a triplet $(a_{i-1}^{\mathcal{T}}, a_i^{\mathcal{T}}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}})$ is indistinguishable from a random one under the random oracle hypothesis as long as he does not make a query to the oracle. The only way to distinguish communications is thus to find at least one collision between $L_{\mathcal{C}}^{b}$ and $L^1 \cup L^2$. The probability to get such a collision is negligible.
- **Case 2:** $\mathcal{A}$ made some random oracle queries but none of the form $H(a_i^{\mathcal{T}}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}} \oplus \hat{K})$ where $\hat{K}$ is a key in S_1 or S_2 and $a_i^{\mathcal{T}}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}}$ is a part of an output from a tag. In this case, $\mathcal{A}$ has no more information than in the previous case, and the conclusion is the same

– **Case 3:** $\mathcal{A}$ made some random oracle queries and one of them is of the form $H(a_i^{\mathcal{T}}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}} \oplus \hat{K})$. This means $\mathcal{A}$ got a key of this system. As we already proved in a previous remark, this event has a negligible probability to happen.

The overall advantage of $\mathcal{A}$ is negligible in the security parameters for a polynomially bounded adversary. □

9 Conclusion

Following a general trend in inserting PUFs inside RFIDs, we modify the Tree-Based Hash protocols to allow the integration of POKs. Because of the fact that keys inside a tag are now physically obfuscated, we show that an adversary is not able to impersonate a tag. Moreover, we prove our tag system has no privacy leakage. We thus believe that our work helps to strengthen the security of the overall protocol.

Acknowledgments

The authors wish to thank the anonymous reviewers for their comments, Jean-Sébastien Coron and Bruno Kindarji for their help to enhance the quality of the paper.

References

1. Auto-ID Center. Draft protocol specification for a 900 MHz Class 0 Radio Frequency Identification Tag (2003)
2. Avoine, G., Buttyán, L., Holczer, T., Vajda, I.: Group-based private authentication. In: Proceedings of the International Workshop on Trust, Security, and Privacy for Ubiquitous Computing (TSPUC 2007). IEEE, Los Alamitos (2007)
3. Bolotnyy, L., Robins, G.: Physically Unclonable Function-based security and privacy in RFID systems. In: International Conference on Pervasive Computing and Communications – PerCom 2007, New York, USA, March 2007, pp. 211–220. IEEE Computer Society Press, Los Alamitos (2007)
4. Buttyán, L., Holczer, T., Vajda, I.: Optimal key-trees for tree-based private authentication. In: Privacy Enhancing Technologies, pp. 332–350 (2006)
5. Damgård, I., Pedersen, M.Ø.: RFID security: Tradeoffs between security and efficiency. In: CT-RSA 2008 (2008)
6. Gassend, B.: Physical random functions. Master's thesis, Computation Structures Group, Computer Science and Artificial Intelligence Laboratory, MIT (2003)
7. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security, pp. 148–160. ACM, New York (2002)
8. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)

9. Juels, A., Weis, S.A.: Defining strong privacy for RFID. In: PERCOMW 2007, pp. 342–347. IEEE Computer Society Press, Washington (2007)
10. Molnar, D., Wagner, D.: Privacy and security in library RFID: issues, practices, and architectures. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 210–219 (2004)
11. Molnar, D., Soppera, A., Wagner, D.: A scalable, delegatable pseudonym protocol enabling ownership transfer of RFID tags. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 276–290. Springer, Heidelberg (2006)
12. Nohara, Y., Inoue, S., Baba, K., Yasuura, H.: Quantitative evaluation of unlinkable id matching systems. In: Workshop on Privacy in the Electronic Society (2006)
13. Nohl, K., Evans, D.: Quantifying information leakage in tree-based hash protocols (short paper). In: ICICS, pp. 228–237 (2006)
14. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: DAC, pp. 9–14. IEEE, Los Alamitos (2007)
15. Tuyls, P., Batina, L.: RFID-tags for anti-counterfeiting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 115–131. Springer, Heidelberg (2006)
16. Vaudenay, S.: On privacy models for RFID. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 68–87. Springer, Heidelberg (2007)

A Security Proofs

Completeness
In our scheme, errors could occur because of collisions in the output of the hash function. For instance, a part of a communication $(a_{i-1}, H(a_{i-1}, r_i^{\mathcal{T}}), r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}})$ could lead to an error when for a tag $\mathcal{T}'$, we get the equality $H(a_{i-1}, r_i^{\mathcal{T}}) = H(a_{i-1}, r_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}} \oplus K_i^{\mathcal{T}'})$. This could appear with a probability at most $\frac{Q}{2^{l_H}}$. Because there are d stages, the overall probability to fail in the identification is $O(\frac{dQ}{2^{l_H}})$ which is negligible in the parameters.

Soundness
We first remark the following. Let us define a family of functions $H_{a,b}$ derived from H. b is a bit string of size l_b. $H_{a,b}$ is a function from $\{0,1\}^{l_b}$ to $\{0,1\}^{l_H}$ such that $H_{a,b}(x) = H(a, b \oplus x)$. As H is preimage resistant, we can consider that an adversary has a negligible probability to find a preimage of $H_{a,b}(x)$ whatever a and b are. If there are polynomially many a_i and b_j, an adversary has a negligible probability to find x even if he knows $H_{a_i,b_j}(x)$ for all a_i and b_j.

Now, we denote L_1 the list of all the communications produced via the SendTag queries, L_2 the communications sent to the TC either with the SendTC query or the Result query.

To simplify the notation, we prove that the scheme is sound with $d = 1$. This is a sufficient condition, as the difficulty to authenticate increases with d. We denote M, the maximum number of operations made by $\mathcal{A}$.

Assume $\mathcal{A}$ has received the challenge a_0^{TC} and outputs the couple (a_1, x_1). We overestimate the probability of success of $\mathcal{A}$. There are two cases:

- **Case 1:** $\mathcal{A}$ did not use H to output a_1. This means:
  - either a_0^{TC} had been tested by $\mathcal{A}$ thanks to the SendTag query, this could arise with a probability less than $\frac{M}{2^{l_r}}$,

Table 1. Resources needed in the first case

Tag			Tag → TC	TC
numbers	non-volatile memory	computation	communication	computation
2^{20}	200 bits	2 AES and 2 random	328 bits	2×2^{10}
2^{30}	300 bits	3 AES and 3 random	492 bits	3×2^{10}

  - or he tried a random answer. In this sub-case, he has a probability of success less than $\frac{M.Q}{2^{l_H}}$ thanks to the collision resistance property.
- **Case 2:** $\mathcal{A}$ used H to output a_1. So we denote $a_1 = H(a_0^{\mathsf{TC}}, x_1')$. This means
  - either there is no key $\hat{K}$ like $x_1' = x_1 \oplus \hat{K}$. The probability of success is thus less than $\frac{M.Q}{2^{l_H}}$,
  - or there is a key $\hat{K}$ in the key material such that $x_1' = x_1 \oplus \hat{K}$. Consequently $\mathcal{A}$ possesses one key. He could achieve this only using the information from L_1 and L_2. $\mathcal{A}$ only knows triplets of the form $\big(a_0, H(a_0, r^{\mathcal{T}}), r^{\mathcal{T}} \oplus K^{\mathcal{T}}\big)$ for some tags $\mathcal{T}$. A change of variable leads to: $a_0, H_{a_0,r^{\mathcal{T}}}(K^{\mathcal{T}}), r^{\mathcal{T}}$. Thanks to the previous remark on preimage resistance property, we can conclude that the probability $\mathcal{A}$ got one key is negligible.

We can conclude that our scheme is sound as the overall probability of any adversary is negligible.

B Practical Example

We propose for our protocol, as an example, the following parameters:

- the size of the reader challenge l_r is 64,
- the size of any POK l_K is 100
- the size of the output of H l_H is 64.
  For instance, the first 64 bits of $AES_{a_{i-1,1..28}||r_i}(a_{i-1}||r_{i,1..64})$.

They have been chosen to minimize the non-volatile memory inside the tag and the communication between tags and readers, but they should lead to a sufficient security to insure the secrecy of the keys and the impossibility to authenticate without the knowledge of the keys. We use AES as it is possible to implement it with not too many gates and because the problem to find a preimage or any collision is usually believed intractable.

Security can be improved by increasing the parameters:

- the size of the reader challenge l_r is 64,
- the size of any POK l_K is 116
- the size of the output of H l_H is 64.
  For instance, the first 64 bits of $AES_{a_{i-1,1..12}||r_i}(a_{i-1}||r_{i,1..64})$.

The two tables Table 1 and Table 2 summarize the concrete resources used in our scheme in the two previous cases for some example parameters. We use a branching factor of 2^{10} in all cases.

Table 2. Resources needed in the second case

Tag			Tag → TC	TC
numbers	non-volatile memory	computation	communication	computation
2^{20}	232 bits	2 AES and 2 random	360 bits	2×2^{10}
2^{30}	348 bits	3 AES and 3 random	540 bits	3×2^{10}

Two Generic Constructions of Probabilistic Cryptosystems and Their Applications

Guilhem Castagnos

GREYC, Ensicaen,
Boulevard Maréchal Juin, BP 5186, 14032 Caen cedex, France
guilhem.castagnos@info.unicaen.fr

Abstract. In this paper, we build, in a generic way, two asymmetric cryptosystems with a careful study of their security. We present first an additively homomorphic scheme which generalizes, among others, the Paillier cryptosystem, and then, another scheme, built from a deterministic trapdoor function. Both schemes are proved semantically secure against chosen plaintext attacks in the standard security model and modify versions can be proved secure against adaptive chosen ciphertext attacks.

By implementing these constructions with quotients of $\mathbf{Z}$, elliptic curves and quadratic fields quotients we get some cryptosystems yet described in the past few years and provide variants that achieve higher levels of security than the original schemes. In particular, using quadratic fields quotients, we show that it is possible to build a new scheme secure against adaptive chosen ciphertext attacks in the standard security model.

Keywords: Probabilistic Encryption, Homomorphic Scheme, Generic Construction, Paillier Cryptosystem, Quadratic Fields, IND-CPA and IND-CCA2 security, Standard Model.

1 Introduction

In 1984, Goldwasser and Micali have designed the first probabilistic cryptosystem and defined the adequate notion of security for this type of scheme: the notion of semantic security. After this system, based on quadratic residuosity, many probabilistic schemes built from the same principle have been proposed: chronologically by Benaloh ([Ben88]), Naccache and Stern ([NS98]), Okamoto and Uchiyama ([OU98]) and at last, the most achieved system have been proposed by Paillier ([Pai99]) and then generalized by Damgård and Jurik (cf. [DJ01]), allowing to encrypt larger messages. All these schemes use quotients of $\mathbf{Z}$, their one-wayness is based on factoring and their semantic security is based on the hardness of distinguishing some powers. Moreover, these schemes are additively homomorphic, *i. e.*, if we got a multiplicative group structure on the ciphertexts set and an additive one on the plaintexts set, then, if c_i is a valid encryption of m_i, with $i \in \{1, 2\}$, $c_1 c_2$ is a valid ciphertext of $m_1 + m_2$. This property has many

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 92–108, 2008.

applications, for example the systems of Paillier and Damgård and Jurik can be used to design electronic vote systems (cf. [BFP+01, Jur03]), for Private Information Retrieval (cf. [Lip05]), or for building Mix-nets (cf. [NSNK06, Jur03]). At the present time, the Paillier and Damgård-Jurik cryptosystems are almost the only schemes that are additively homomorphic and practical. The system of Paillier has also been adapted in elliptic curves over $\mathbf{Z}/n^2\mathbf{Z}$ by Galbraith in [Gal02]. Another finite group, simpler than elliptic curves over finite ring can be used to adapt this system: the group of norm 1 quadratic integers modulo n, where n is an RSA integer (this adaptation was only briefly sketched in [Cas07]).

A fast and non-homomorphic variant of the Paillier scheme has been proposed by Catalano, Gennaro *et al.* in [CGH+01], and later adapted in elliptic curves by Galindo, Martín *et al.* (cf. [GMMV03]) and again in quadratic fields quotients in [Cas07]. These schemes can also be seen like probabilistic variants of deterministic trapdoor functions: respectively RSA, KMOV (cf. [KMOV92]) and LUC (cf. [SL93]).

In this paper, we propose two generic constructions that capture the ideas of all these schemes. In section 2, we show how to build a generic homomorphic encryption trapdoor whose semantic security is based on the hardness of the problem of distinguishing k^{th} powers of a group, for a well-chosen integer k. Note that this construction is essentially known as it is a direct generalization of the Paillier scheme. We include it here for completeness as a formal exposition is not known by the author. Then, in section 3, we modify the previous construction in order to get more efficient schemes. This will result in a method to build a probabilistic trapdoor function from a deterministic trapdoor function which satisfies some properties.

For each construction, we do a careful study of both one-wayness and semantic security. For the first one, we begin with a scheme secure against chosen-plaintext attacks (the homomorphic schemes can not be secure against chosen-ciphertext attacks because of their obvious malleability) and then we show that we can modify this construction to use universal hash proof systems (cf. [CS02]) in order to build an IND-CCA2 scheme in the standard model. The second construction can be viewed as a simple way to transform a deterministic trapdoor function into an encryption primitive IND-CPA secure in the standard model against a decision problem relative to the properties of the deterministic trapdoor function used. We also present a variant IND-CCA2 secure in the random oracle model by using standard techniques.

In section 4, we apply these generic constructions in quotients of $\mathbf{Z}$, elliptic curves and quadratic fields quotients. By doing this, we will see that a large number of probabilistic schemes proposed these last years can be considered as applications of the generic constructions. This study also leads to an historical treatment of probabilistic encryption based on factoring. With quadratic fields quotients, the application of the generic construction of section 2 leads to a concise but detailed description of the practical homomorphic cryptosystem only briefly sketched at the end of [Cas07]. Moreover, we will show that this scheme can be transformed to build an IND-CCA2 secure cryptosystem in the standard model.

Notations: In all the paper, G will denote a finite multiplicative abelian group, k a nonnegative integer and g an element of G of order k. We will denote $|G|$ the order of the group G. Let G^k be the subgroup of k^{th} power of G. We will suppose that $k \,\big|\, |G|$ and denote $\lambda := |G|/k$. Moreover, we will suppose that λ and k are coprime. Given a group element h, $\langle h \rangle$ will denote the group generated by h.

Given an integer i, $|i|_2$ will denote the size of i in bits, *i. e.*, $|i|_2 := \lfloor \log_2 k \rfloor + 1$.

We will denote by n an RSA integer, *i. e.*, n will be the product of two distinct odd primes p and q, large enough, such that the factorization of n is infeasible in reasonable time (*i. e.*, $|n|_2 \geq 1024$).

For two algorithmic problems A and B, we will denote $A \overset{\mathcal{P}}{\Longleftarrow} B$ whenever A is polynomial-time reducible to B, and $A \overset{\mathcal{P}}{\Longleftrightarrow} B$ whenever the two problems are polynomial-time equivalent.

2 Additively Homomorphic Trapdoor Function

Let us first state a straightforward result of group theory.

Theorem 1. *Let G be a finite multiplicative abelian group, k a nonnegative integer such that k divides $|G|$ and that k and $\lambda := |G|/k$ are coprime, then*

1. *the order of G^k is λ;*
2. *the order of the quotient group G/G^k is k;*
3. *$G^k = \{x \in G,\ x^\lambda = 1\}$;*
4. *If g is an element of G of order k then G/G^k is cyclic and $G/G^k = \langle \pi(g) \rangle$ where π denotes the canonic surjection $\pi : G \to G/G^k$.*

Proof (sketch). We use the decomposition of G in a direct sum of cyclic groups, and the fact that in a cyclic group of order n, the equation $x^k = 1$ has zero or $\gcd(n, k)$ roots. As a consequence, there are k k^{th} roots of unity in G and the kernel of the map $x \mapsto x^k$ has order k. This proves 1. and 2.; to prove 3. and 4., one uses the fact that λ and k are coprime. □

From this theorem, one can also deduce that G^λ has order k and that G^λ is actually the subgroup of k^{th} roots of unity of G. Note that g will be a generator of G^λ, *i. e.*, $G^\lambda = \langle g \rangle$. One can see that there is an isomorphism:

$$G^\lambda \times G^k \xrightarrow{\sim} G.$$

The evaluation of this isomorphism if easy: one simply multiply the two elements. The decomposition of an element of G in a product of a k^{th} root of unity by a k^{th} power is less obvious, unless one knows the values of λ and k. As these integers are coprime, there exists μ and ν such that $\mu\lambda + \nu k = 1$ and $c = \left(c^\mu\right)^\lambda \left(c^\nu\right)^k$. In the following, we are going to use this isomorphism to build the trapdoor function. Before that, we define a decision problem.

Definition 1. *We will call the decision residuosity problem of degree k in G, and will denote* $\mathrm{Res}_{G,k,g}$*, the following problem: Given c an element of G and g an element of order k*[1]*, decide whether $c \in G^k$ or not.*

We want to build an homomorphic encryption whose semantic security is based on the difficulty of the decision residuosity problem of degree k in G. This construction will generalize, among others, the system of Paillier (cf. [Pai99]) where $G = \left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$ with n an RSA integer and $k = n$.

Public Key. The group G, the integer k and the element g will be public. Plaintext messages will be the elements of $\mathbf{Z}/k\mathbf{Z}$. We will suppose known an efficient algorithm to generate random elements of G^k, and an efficient algorithm to compute the discrete logarithm to base g in $\langle g \rangle$.

Encryption Primitive:

$$\mathcal{E}_{G,k,g} \ : \ \begin{cases} \mathbf{Z}/k\mathbf{Z} \longrightarrow G \\ m \quad \longmapsto g^m \rho \end{cases}$$

where ρ is a random element of G^k.

According to Theorem 1, 4., if π denotes the canonic surjection $\pi : G \to G/G^k$, $\pi(g)$ is a generator of the quotient group G/G^k. So if $c \in G$ is an encryption of $m \in \mathbf{Z}/k\mathbf{Z}$, we will have

$$m = \log_{\pi(g)}\left(\pi(c)\right).$$

As a consequence, the decryption function associated to $\mathcal{E}_{G,k,g}$ will be a surjective morphism from $(G, \times)$ to $(\mathbf{Z}/k\mathbf{Z}, +)$, and a cryptosystem based on the $\mathcal{E}_{G,k,g}$ primitive will be additively homomorphic. As the scheme is homomorphic, it also enjoys the "self-blinding" property: given c an encryption of m, one can produce another valid ciphertext c' of m by computing $c' := c\rho'$, where ρ' is a random element of G^k.

Private Key and Decryption Algorithm. The integer λ is a trapdoor for the $\mathcal{E}_{G,k,g}$ function. Let $c \leftarrow \mathcal{E}_{G,k,g}(m)$. There exists an element $\rho \in G^k$ such that $c = g^m\rho$. According to Theorem 1, 3., $c^\lambda = g^{m\lambda}$. Thanks to the public algorithm for the discrete logarithm problem in $\langle g \rangle$, we can recover $m\lambda$ in the ring $\mathbf{Z}/k\mathbf{Z}$, and them m, as λ and k are coprime.

One-Wayness. Let us define a new computational problem.

Definition 2. *Given c an element of G we will call the residuosity class of degree k of c the element m of $\mathbf{Z}/k\mathbf{Z}$ such that $m = \log_{\pi(g)}\left(\pi(c)\right)$. We will denote* $\mathrm{Class}_{G,k,g}$*, the problem of computing the residuosity class of degree k of elements of G.*

[1] This condition is technical, in order to prove the equivalence in Theorem 3. We will see that in practice, (cf. section 4), given G and k, it will be easy to find an element of order k in G.

A scheme built from the $\mathcal{E}_{G,k,g}$ function will be one-way if and only if the $\text{Class}_{G,k,g}$ problem is hard. It is easy to see that this problem is random self-reducible (so all the instances of the problem have the same complexity) and does not depend of the choice of the element g of order k, thanks to the properties of the discrete logarithm.

In the decryption algorithm, we have seen that one can decrypt an encryption $c = g^m \rho$ of m thanks to the knowledge of λ. It is also possible to decrypt c by computing the element x of G such that $x^k \equiv c \pmod{G^\lambda}$. Note that x is indeed unique modulo $G^\lambda = \langle g \rangle$, the subgroup of k^{th} roots of unity. As

$$m = \log_{\pi(g)} \left(\pi(c)\right) = \log_{\pi(g)} \left(\pi(c/x^k)\right),$$

and as c/x^k is an element of $G^\lambda = \langle g \rangle$, one can recover m by computing the discrete logarithm of c/x^k to base g in $\langle g \rangle$. As a consequence of the existence of this decryption process, we define another computational problem in order to analyse the $\text{Class}_{G,k,g}$ problem.

Definition 3. *We will denote* $\text{C–RSA}_{G,k}$*, the following problem: Given* c *an element of* G*, find* x *such that* $x^k \equiv c \pmod{G^\lambda}$.

Remark 1. If one knows how to manipulate the elements of G/G^λ and to lift them in G, the $\text{C–RSA}_{G,k}$ problem is equivalent to the problem of the local inversion of the automorphism $x \mapsto x^k$ of G/G^λ, which is a generalization of the RSA function (G/G^λ has order λ which is prime to the exponent k).

If one knows λ, *i. e.*, the order of G, one can solve the $\text{C–RSA}_{G,k}$ problem: given $c \in G$, the element

$$x := c^{k^{-1} \mod \lambda}$$

verifies $x^k \equiv c \pmod{G^\lambda}$. As a consequence, we can state the following theorem which generalizes Theorem 1 and 2 of [Pai99].

Theorem 2. *Let* G *be a finite multiplicative abelian group,* k *a nonnegative integer such that* k *divides* $|G|$ *and that* k *and* $|G|/k$ *are coprime, and let* g *be an element of* G *of order* k*. We have the following reductions:*

$$\text{Class}_{G,k,g} \overset{\mathcal{P}}{\Longleftarrow} \left(\text{C–RSA}_{G,k} \wedge \text{Dlog}_{\langle g \rangle}\right) \overset{\mathcal{P}}{\Longleftarrow} \left(\text{Order}_G \wedge \text{Dlog}_{\langle g \rangle}\right).$$

where $\text{Dlog}_{\langle g \rangle}$ *denotes the discrete logarithm problem in* $\langle g \rangle$ *and* Order_G *the problem of computing* $|G|$*.*

Remark 2. The problem $\text{Dlog}_{\langle g \rangle}$ appears in the previous theorem for completeness, but in practice, as we said earlier, we will hope that this problem is easy in order to be able to decrypt efficiently.

Semantic Security

Theorem 3. *Let* G *be a finite multiplicative abelian group,* k *a nonnegative integer such that* k *divides* $|G|$ *and that* k *and* $|G|/k$ *are coprime, and let* g *be*

an element of G of order k. An encryption scheme built from the $\mathcal{E}_{G,k,g}$ primitive is semantically secure against Chosen-Plaintext Attacks if and only there exists no polynomial algorithm to solve the decision residuosity problem of degree k in G.

Proof. To prove that a scheme is semantically secure, one can use the "real or random property": *i. e.*, prove that no polynomial time algorithm can distinguish an encryption of a chosen message, m, from an encryption of a random message. In our construction, an encryption of a random message is a random element of G. So we have to distinguish a random element of G from an encryption of m. As the scheme is homomorphic, this is equivalent to distinguish encryption of 0 in G, that is an element of G^k, in G. □

Generation of Random Elements of G^k. To generate random elements of G^k, one can just take at random elements of G and raise them to the power of k. If one can work in the quotient group G/G^λ and lift the elements of this group in G, one can also use the isomorphism $G/G^\lambda \to G^k$, $x \mapsto x^k$. The encryption function becomes:

$$\mathcal{E}'_{G,k,g} \; : \; \begin{cases} \mathbf{Z}/k\mathbf{Z} \times G/G^\lambda & \xrightarrow{\sim} & G \\ (m \quad , \quad \rho) & \longmapsto & g^m \rho^k \end{cases}$$

It is trivial to see that $\mathcal{E}'_{G,k,g}$ is a group isomorphism.

Remark 3. If one can not generate random elements of G or random elements of G/G^λ, a solution to generate elements of G^k is to publish an element ρ of G^k of high order and to generate others k^{th} powers by raising ρ to a random power. Note that in this case, the semantic security of the scheme relies on a slightly different problem: the decision problem of distinguishing the elements of $\langle \rho \rangle$ in G.

IND-CCA2 **Variant in the Standard Model.** The system of Paillier, generalized by the previous construction, has been used in [CS02] to build an IND-CCA2 cryptosystem in the standard security model by an application of a general framework built from a subset membership problem and some projective hash families. Our construction with the decision residuosity problem can be easily adapted to fit the framework of [CS02] with only one extra hypothesis. We refer the reader to [CS02] for definitions.

Suppose that the group G is cyclic. Denote $\mathcal{H} = \mathrm{Hom}(G, G)$. Then, from the example 7.4.2 in [CS02], one can prove that the group system $\mathbf{G} := (\mathcal{H}, G, G^k, G)$ is diverse and that the projective hash family derived from $\mathbf{G}$ is $1/\tilde{p}$-universal where $\tilde{p}$ is the smallest prime dividing λ (Theorem 2 of [CS02]). With this, we get an $1/\tilde{p}$-universal hash proof system (UHPFS). Following the general construction of [CS02], from this UHPFS, one can build a scheme that is IND-CCA2 secure in the standard model, providing that $\tilde{p}$ is sufficiently large, and assuming the hardness of the decision residuosity problem.

3 Non-homomorphic Trapdoor Function

In this section, we change the previous construction in order to reduce the encryption and decryption costs. The idea is to replace the most costly step of the encryption process: the evaluation of the function $x \mapsto x^k$. This exponentiation will be replaced by a function f, cheaper to evaluate. This idea corresponds to the scheme of [CGH+01], which uses a function built from the RSA function. By doing this, we will loose the homomorphic property.

We have to build the function f in order to still have an efficient way to decrypt. In the previous section, we saw that if was possible to decrypt by inverting the automorphism $x \mapsto x^k$ of $G/\langle g \rangle$. We are going to replace this automorphism by a known determinist trapdoor function $\overline{f}$, permutation of a subset of $G/\langle g \rangle$. The function f will be built from $\overline{f}$. As a consequence, the construction of this section will enable oneself to build a probabilistic trapdoor function from a determinist one.

Construction of f. We suppose that we know a trapdoor permutation $\overline{f}$ of a subset $\overline{\Lambda}$ of $G/\langle g \rangle$. In this section, π will denote the canonical surjection $G \to G/\langle g \rangle$. We suppose that π is computable at low cost for anyone who knows G and g.

We define $\Omega := \pi^{-1}(\overline{\Lambda})$ and Λ, a subset of Ω such that Λ be a representative set of $\overline{\Lambda}$, *i. e.*, $\pi(\Lambda) = \pi(\Omega) = \overline{\Lambda}$ and π is a bijection from Λ to $\overline{\Lambda}$. We suppose that it is easy to find the unique representative of a class of $\overline{\Lambda}$ in Λ. Let f be a function from Λ to Ω such that the following diagram commutes:

$$\begin{array}{ccccc}
\Lambda & \xrightarrow{\ f\ } & \Omega & \hookrightarrow & G \\
\downarrow{\scriptstyle \wr} & \circlearrowright & \downarrow & & \downarrow{\scriptstyle \pi} \\
\overline{\Lambda} & \xrightarrow[\overline{f}]{\sim} & \overline{\Lambda} & \hookrightarrow & G/\langle g \rangle
\end{array}$$

Public Key. The group G, the integer k and the element g will be public. Plaintext messages will be the elements of $\mathbf{Z}/k\mathbf{Z}$. We will suppose known an efficient algorithm that returns random elements of Λ, an efficient algorithm to evaluate the function f, and an efficient algorithm to compute the discrete logarithm to base g in $\langle g \rangle$.

Encryption Primitive:

$$\mathcal{E}_{G,f,g} \ : \ \begin{cases} \mathbf{Z}/k\mathbf{Z} \times \Lambda \longrightarrow \Omega \\ (m \ \ , \ \rho) \longmapsto g^m f(\rho) \end{cases}$$

It is easy to see that $\mathcal{E}_{G,f,g}$ is well defined as $\langle g \rangle f(\Lambda) = \Omega$ and bijective: suppose that $g^{m_1} f(\rho_1) = g^{m_2} f(\rho_2)$ then $\pi\big(f(\rho_1)\big) = \pi\big(f(\rho_2)\big)$. As $\pi \circ f = \overline{f} \circ \pi$, $\pi \circ f$ is bijective so $\rho_1 = \rho_2$. As a consequence, $m_1 = m_2$ in $\mathbf{Z}/k\mathbf{Z}$.

Private Key and Decryption Algorithm. The private key is the trapdoor that allows to invert $\overline{f}$. Let $c \in \Omega$ be a ciphertext. To decrypt c, we have to recover $m \in \mathbf{Z}/k\mathbf{Z}$ such that there exists $\rho \in \Lambda$ such that $c = g^m f(\rho)$. We have $\pi(c) = \pi \circ f(\rho) = \overline{f} \circ \pi(\rho)$. With the private key we recover $\pi(\rho)$ and then its representative $\rho \in \Lambda$. Then, by computing $c/f(\rho)$ we get g^m and then m thanks to the algorithm for the discrete logarithm problem in $\langle g \rangle$.

One-Wayness. Let us give the definition of the problem on which relies the one-wayness of a scheme built from the $\mathcal{E}_{G,f,g}$ primitive.

Definition 4. *We will denote* $\mathrm{Class}_{G,f,g}$ *the following problem: given* c *an element of* Ω*, find* $m \in \mathbf{Z}/k\mathbf{Z}$ *such that there exists* ρ *in* Λ *such that* $c = g^m f(\rho)$.

Now we define two others problems and we give a theorem that links the three problems.

Definition 5. *We will denote* $\mathrm{Hensel}_{G,g}{-}f$ *the following problem: given* $\overline{c}$ *an element of* $\overline{\Lambda} = \pi(\Omega)$*, find the element* c *of* Ω *such that* $c = f(\rho)$ *where* ρ *is the element of* Λ *such that* $\overline{c} = \pi(f(\rho))$*. We will denote* $\mathrm{Inv} - \overline{f}$ *the problem of local inversion of the trapdoor* $\overline{f}$*,* i. e.*, given* $\overline{c}$ *an element of* $\overline{\Lambda}$*, find* $\overline{\rho}$ *in* $\overline{\Lambda}$ *such that* $\overline{c} = \overline{f}\,(\overline{\rho})$.

Theorem 4. *Let* G *be a finite multiplicative abelian group,* k *a nonnegative integer,* g *an element of* G *of order* k*,* $\overline{\Lambda}$ *a subset of* $G/\langle g \rangle$*,* Λ *a representative set of* $\overline{\Lambda}$ *in* G *and* $\overline{f}$ *a trapdoor permutation of* $\overline{\Lambda}$*. We denote* π *the canonic surjection from* G *to* $G/\langle g \rangle$ *and* f *a function from* Λ *to* $\Omega := \pi^{-1}(\overline{\Lambda})$ *such that* $\pi \circ f = \overline{f} \circ \pi$*. We have the following relations:*

$$\mathrm{Class}_{G,f,g} \overset{\mathcal{P}}{\Longleftrightarrow} \left(\mathrm{Hensel}_{G,g}{-}f \ \wedge \ \mathrm{Dlog}_{\langle g \rangle}\right) \overset{\mathcal{P}}{\Longleftarrow} \left(\mathrm{Inv} - \overline{f} \ \wedge \ \mathrm{Dlog}_{\langle g \rangle}\right)$$

where $\mathrm{Dlog}_{\langle g \rangle}$ *denotes the discrete logarithm problem in* $\langle g \rangle$.

Proof. We prove the left equivalence, the reduction on the right will follow from the decryption algorithm. Suppose that we have two oracles that solve respectively the $\mathrm{Hensel}_{G,g}{-}f$ and $\mathrm{Dlog}_{\langle g \rangle}$ problems. Let c be an element of Ω. We want to recover $m \in \mathbf{Z}/k\mathbf{Z}$ in the decomposition $c = g^m f(\rho)$ with $\rho \in \Lambda$. We have $\pi(c) = \pi\big(f(\rho)\big)$. We give $\pi(c)$ to the oracle for the $\mathrm{Hensel}_{G,g}{-}f$ problem. We get the element c' of Ω such that $c' = f(\rho)$. Given c/c', the oracle for the $\mathrm{Dlog}_{\langle g \rangle}$ problem returns m.

For the opposite way, we have an oracle that solve the $\mathrm{Class}_{G,f,g}$ problem. If g' is an element of $\langle g \rangle$, we take a random element ρ in Λ. By giving $g' f(\rho)$ to the oracle, we get m, the discrete logarithm of g' to base g. Suppose now that we have an element $\overline{c}$, of $\overline{\Lambda}$, for which we want to solve the $\mathrm{Hensel}_{G,g}{-}f$ problem. We take m' at random in $\mathbf{Z}/k\mathbf{Z}$. We denote c the element of Λ such that $\pi(c) = \overline{c}$. We give $g^{m'} c \in \Omega$ to the oracle (note that it is a random query for the oracle). We then get from the oracle the element m of $\mathbf{Z}/k\mathbf{Z}$ such that $g^{m'} c = g^m f(\rho)$ with ρ element of Λ. As $\pi(g^{m'} c) = \overline{c} = \pi\big(f(\rho)\big)$, the element $g^{m'-m} c$ is a correct answer to the $\mathrm{Hensel}_{G,g}{-}f$ problem. □

Remark 4. This theorem establishes that the security of a system built from the $\mathcal{E}_{G,f,g}$ primitive relies on the security of the trapdoor $\overline{f}$. For the Catalano *et al.* scheme, (cf. [CGH+01] and section 4), an instance of this construction in which $\overline{f}$ is the classic RSA function, the result of [CNS02] states that the equivalence actually holds. Unfortunately, the proof of this result uses intrinsic properties of the RSA function and can not be exploited for the generalized case.

Semantic Security

Definition 6. *Let us denote* $\mathrm{Res}_{G,f,g}$*, the problem of distinguishing the elements of* $f(\Lambda)$ *in* Ω*.*

Theorem 5. *An encryption scheme built from the* $\mathcal{E}_{G,f,g}$ *primitive is semantically secure against Chosen-Plaintext Attacks if and only there exists no polynomial algorithm that solve the decision* $\mathrm{Res}_{G,f,g}$ *problem.*

Proof. A scheme built from the construction of the previous section, and a scheme built from $\mathcal{E}_{G,f,g}$ shares a similar property:

$$(c \leftarrow \mathcal{E}_{G,f,g}(m)) \iff \left(\frac{c}{g^m} \in f(\Lambda)\right).$$

As a consequence, the proof of Theorem 3 can be easily adapted. □

IND-CCA2 **Variant in the** ROM**.** Using standard techniques, one can modify the $\mathcal{E}_{G,f,g}$ primitive to make it resistant against adaptive chosen-ciphertext attacks in the random oracle model. One can simply add $h(m,\rho)$ to the ciphertext, where h is an hash function viewed like a random oracle. One can also use the Fujisaki-Okamoto conversion (cf. [FO99]) in order to reduce the ciphertexts size.

4 Applications

We will use the constructions of sections 2 and 3 in algebraic groups over $(\mathbf{Z}/n^s\mathbf{Z})^\times$ where s is a nonnegative integer. RSA integers will allow to use the group order as a trapdoor. This would lead to an historical of probabilistic cryptography based on factoring.

The idea of working modulo n^s with $s > 1$ is due to Paillier (cf. [Pai99]) and Damgård and Jurik (cf. [DJ01]) for the case $s > 2$. As we shall see in the following, this enables oneself to meet the hypothesis of the generic construction: the subgroup of n^{th} roots of unity of the group considered will be the kernel of the reduction modulo n, and its elements will be easy to describe. As a consequence, we will exhibit an element g of order n such that the discrete logarithm problem in $\langle g\rangle$ is easy.

4.1 Schemes in Quotients of Z

The first probabilistic cryptosystem, proposed by Goldwasser and Micali in 1984 (cf. [GM84]) is very similar to the generic construction explained in section 2. Its semantic security is based on a well-known problem, the quadratic residuosity problem (*i. e.*, $k = 2$), but its expansion is awful as one bit is encrypted with $|n|_2$ bits.

$G = (\mathbf{Z}/n\mathbf{Z})^\times$, k Prime, $k \mid \varphi(n)$, Benaloh (88). The cryptosystem of Goldwasser-Micali has been generalized by Benaloh in [Ben88]. The group G is now $(\mathbf{Z}/n\mathbf{Z})^\times$, the integer k is an odd prime such that k divides $\varphi(n)$ and k does not divide $\lambda := \varphi(n)/k$. Let g be an element of order k, to encrypt an element $m \in \mathbf{Z}/k\mathbf{Z}$, one uses the encryption primitive $\mathcal{E}_{G,k,g}$ defined in section 2: an encryption of m is $g^m r^k$ where r is a random element of $(\mathbf{Z}/n\mathbf{Z})^\times$. The drawback of this system is that k has to be small because there is no particular algorithm for computing discrete logarithms in $\langle g \rangle$. As a consequence, the expansion of the system, $|n|_2 / |k|_2$ remains high.

$G = (\mathbf{Z}/n\mathbf{Z})^\times$, k Smooth, $k \mid \varphi(n)$, Naccache-Stern (98). Naccache and Stern have improved in [NS98] the previous system. They still use $G = (\mathbf{Z}/n\mathbf{Z})^\times$ but k is chosen smooth. This leads to a more efficient algorithm for computing discrete logarithms in $\langle g \rangle$ by using the Pohlig-Hellman algorithm. Naccache and Stern state that the expansion can be reduced to 4.

Okamoto and Uchiyama have proposed in [OU98] to work modulo $n = p^2 q$. The following system is an improvement of their proposal.

$G = \left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$, $k = n$, Paillier (99). The system of Paillier (cf. [Pai99]) corresponds to an application of the $\mathcal{E}_{G,k,g}$ encryption function with $G = \left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$, and $k = n$. If we suppose that $\gcd(n, \varphi(n)) = 1$, as $|G| = n\varphi(n)$, k divides $|G|$ and k is prime to $\lambda := |G|/k = \varphi(n)$. One can see that the subgroup G^λ of G, the subgroup of n^{th} roots of unity, is the kernel of the surjective homomorphism: $\left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times \to (\mathbf{Z}/n\mathbf{Z})^\times$. As a consequence, this subgroup is a cyclic group of order n, generated by $g :\equiv 1 + n \pmod{n^2}$. Moreover, the discrete logarithm problem in $\langle g \rangle$ is trivial as for all $i \in \mathbf{Z}/n\mathbf{Z}$, $g^i \equiv 1 + in \pmod{n^2}$. To encrypt, one can use the isomorphism $\mathcal{E}'_{G,k,g}$ defined in section 2. The encryption function is thus the isomorphism:

$$\begin{cases} \mathbf{Z}/n\mathbf{Z} \times (\mathbf{Z}/n\mathbf{Z})^\times & \xrightarrow{\sim} & \left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times \\ (m\ ,\ r) & \longmapsto & g^m r^n \end{cases}$$

where m is the plaintext and r a random element. The trapdoor is $\varphi(n)$, *i. e.*, the factorization of n, and the decryption algorithm is the application of the generic algorithm described in section 2. The expansion of this system is 2.

An IND-CCA2 variant of this scheme has been designed by Cramer and Shoup in [CS02]. As previously said, this variant can also be obtained from the construction of section 2, if the group G is cyclic. One can have a cyclic group by choosing Sophie Germain primes for p and q: with this choice there exists a cyclic group of order $n\varphi(n)/2$ in $\left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$, isomorphic to $\mathbf{Z}/n\mathbf{Z} \times (\mathbf{Z}/n\mathbf{Z})^+$, where $(\mathbf{Z}/n\mathbf{Z})^+$ is the subgroup of elements of $\left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$ that have a positive Jacobi symbol (see [CS02] subsection 8.2 for details).

Damgård and Jurik have proposed in [DJ01] a generalization of the Paillier cryptosystem. They work in the group $G = \left(\mathbf{Z}/n^{s+1}\mathbf{Z}\right)^\times$ with $s > 1$ and $k = n^s$. One obtains a system that allows oneself to encrypt messages of arbitrary

length (by increasing s). This can have many applications (cf. [DJ01, Jur03]). The expansion of this scheme is $1 + 1/s$.

***G* = $(\mathbf{Z}/n^2\mathbf{Z})^\times$, $\overline{f}$ = *RSA*, Catalano *et al.* (01).** In [CGH+01], Catalano, Gennaro *et al.* have proposed a probabilistic encryption scheme presented like a fast variant of the Paillier cryptosystem. With the help of the generic construction of section 3, one can also see this scheme as a probabilistic version of the RSA cryptosystem. Let $G = \left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$, and $g \equiv 1 + n \pmod{n^2}$. The quotient group $G/\langle g \rangle$ is isomorphic to $(\mathbf{Z}/n\mathbf{Z})^\times$. We denote respectively Ω and Λ, the sets of elements of G and $G/\langle g \rangle$, *i. e.*,

$$\Omega := \left\{ r \in \mathbf{N},\, 0 < r < n^2,\, \gcd(r, n) = 1 \right\},$$

and

$$\Lambda := \left\{ r \in \mathbf{N},\, 0 < r < n,\, \gcd(r, n) = 1 \right\}.$$

With the notation of section 3, one actually has $\overline{\Lambda} := \Lambda$, and the set Λ is a representative set of the classes of Ω modulo n. Let e be an integer prime to $\varphi(n)$, the RSA function, $\overline{f} : x \mapsto (x^e \bmod n)$ is a permutation of Λ. This function is lifted from Λ to Ω by considering $f : x \mapsto (x^e \bmod n^2)$, so that $\pi \circ f = \overline{f} \circ \pi$. To encrypt, we use the $\mathcal{E}_{G,f,g}$ primitive and we obtain the following encryption function:

$$\begin{cases} \mathbf{Z}/n\mathbf{Z} \times \Lambda \longrightarrow & \Omega \\ (m \quad , r) \longmapsto & g^m r^e \bmod n^2 \end{cases}$$

where m is the plaintext and r a random element. The decryption is done has described in section 3: one reduces the ciphertext modulo n and recover r by inverting the RSA function, thanks to the knowledge of d, the inverse of e modulo $\varphi(n)$, the trapdoor of the function $\overline{f}$.

Remark 5. The previous scheme can be generalized by taking $G = \left(\mathbf{Z}/n^{s+1}\mathbf{Z}\right)^\times$ with $s > 1$, in order to decrease the expansion. One has to redefine the set Ω accordingly and to lift $\overline{f}$ in $f : x \mapsto x^e \bmod n^{s+1}$.

One can apply the non-homomorphic construction of section 3, with all the known trapdoor functions of $\mathbf{Z}/n\mathbf{Z}$, *e. g.*, Demytko's (cf. [Dem94]) or LUC (cf. [SL93]). Note that with the LUC function, one gets a scheme already proposed in [Cas07].

4.2 Schemes in Elliptic Curves over $\mathbf{Z}/n^{s+1}\mathbf{Z}$

Both constructions can be applied in elliptic curves. This leads respectively to the systems of Galbraith (cf. [Gal02]) and Galindo, Martín *et al.* (cf. [GMMV03]).

***G* = *E*/($\mathbf{Z}/n^{s+1}\mathbf{Z}$), *k* = n^s, Galbraith (02).** In [Gal02], Galbraith has adapted the Damgård and Jurik scheme (and hence the Paillier scheme) in elliptic curves. This homomorphic scheme can also be viewed as an application of

the $\mathcal{E}_{G,k,g}$ primitive of section 2. The group G is the group of points of an elliptic curve over $\mathbf{Z}/n^{s+1}\mathbf{Z}$, *i. e.*, the set of elements $(X : Y : Z)$ of $\mathbf{P}^2(\mathbf{Z}/n^{s+1}\mathbf{Z})$ such that

$$Y^2Z = X^3 + aXZ^2 + bZ^3,$$

where a and b are two elements of $\mathbf{Z}/n^{s+1}\mathbf{Z}$ such that $4a^3 + 27b^2$ is invertible. We denote this group $E_{a,b}/(\mathbf{Z}/n^{s+1}\mathbf{Z})$ (See [Gal02] for more details on elliptic curves over rings).

One can prove that the order of this group is $n^s\,|E_{a,b}/(\mathbf{Z}/n\mathbf{Z})|$. By taking $k = n^s$, and supposing that n^s is prime to $\lambda := |E_{a,b}/(\mathbf{Z}/n\mathbf{Z})|$, one can apply the generic construction. The tricky part of this adaptation is to find an element g of G of order n^s such that the discrete logarithm problem is easy in $\langle g\rangle$. Once again, we look for g in the kernel of the reduction modulo n from $E_{a,b}/(\mathbf{Z}/n^{s+1}\mathbf{Z})$ to $E_{a,b}/(\mathbf{Z}/n\mathbf{Z})$. One can see that the element $g := (n : 1 : n^3 + an^7 + bn^9 + \cdots)$ is of order n^s and that discrete logarithms are easy to compute in $\langle g\rangle$ (again see [Gal02] for details on this element g, on the subgroup $\langle g\rangle$ and how to compute the group law in this subgroup and in G).

To encrypt a message m of $\mathbf{Z}/n^s\mathbf{Z}$, one use the $\mathcal{E}_{G,k,g}$ primitive of section 2: a ciphertext for m is a point of the form $m.g + P$ where P is a random "$n^{s^{\text{th}}}$ power". To produce a such P, as it is difficult to produce an element of the curve without knowing the factorization of n, one can not take a random element of G or of $E_{a,b}/(\mathbf{Z}/n\mathbf{Z})$ and take it to the "power" n^s. Hence, we use the method exposed in Remark 3: a $n^{s^{\text{th}}}$ power is part of the public key.

A drawback of this scheme is its cost as one has to do costly scalar multiplications in elliptic curve over a huge base ring (as the security is based on factorization and not on the discrete logarithm problem, we can not reduce the size of this ring).

***G = E/(Z/n²Z), $\overline{f}$ = KMOV*, Galindo *et al.* (03).** In [GMMV03], Galindo, Martín *et al.* have proposed a non-homomorphic scheme based on the KMOV trapdoor permutation (cf. [KMOV92]). This scheme is not a direct adaptation of the generic construction of section 3 as the KMOV function is not a permutation of a subset of a group. Indeed, the KMOV function is a permutation of the set

$$\left\{(x, y) \in \mathbf{Z}/n\mathbf{Z} \times \mathbf{Z}/n\mathbf{Z},\ \left(y^2 - x^3\right) \in (\mathbf{Z}/n\mathbf{Z})^\times\right\},$$

and maps (x, y) to $e.(x, y)$, where the scalar multiplication is performed on the elliptic curve $E_{0,y^2-x^3}/(\mathbf{Z}/n\mathbf{Z})$ where e is prime to $(p + 1)(q + 1)$ and p and q are chosen congruent to 2 modulo 3 (it is hard to take points on a fixed curve without knowing p and q). So, one has to apply the generic construction with a group G that depends on the plaintext message. One define *ad hoc* subsets Λ and Ω of $\mathbf{Z}/n^2\mathbf{Z}$ and lift the KMOV function from Λ to Ω by computing $e.(x, y)$ in a curve modulo n^2. See [GMMV03] for more details.

Again, one can generalize this scheme by working modulo n^s with $s > 2$.

4.3 Additively Homomorphic Scheme in Quadratic Fields Quotients

In this subsection, we apply the generic construction of section 2 in another finite group, not widely used in cryptography, the group of norm 1 elements of a quadratic field modulo n. We will obtain the system only briefly sketched at the end of [Cas07].

Definition 7. *Let Δ be a non-square integer, and a an odd integer prime to Δ. We will denote $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$ the group of norm one elements of $\mathcal{O}_\Delta/a\mathcal{O}_\Delta$, where $\mathcal{O}_\Delta$ denotes the ring of integers of $\mathbf{Q}(\sqrt{\Delta})$. We will denote $\varphi_\Delta(a)$ the order of the group $(\mathcal{O}_\Delta/a\mathcal{O}_\Delta)^\wedge$.*

We refer the reader to [Cas07] for the basic properties of this group. We only recall that exponentiation can be efficiently computed in this group by using the Lucas sequence, and that if n is prime to Δ, then for $s > 1$, the order of $\left(\mathcal{O}_\Delta/n^{s+1}\mathcal{O}_\Delta\right)^\wedge$ is

$$\varphi_\Delta(n^{s+1}) = n^s\,\varphi_\Delta(n) = n^s\left(p - \left(\frac{\Delta}{p}\right)\right)\left(q - \left(\frac{\Delta}{q}\right)\right),$$

where $\left(\frac{\Delta}{p}\right)$ denotes the well-known Legendre symbol. Moreover, note that the group $(\mathcal{O}_\Delta/p^s\mathcal{O}_\Delta)^\wedge$ is cyclic (the same holds modulo q^s).

$G = \left(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta\right)^\wedge$, $k = n$,. We apply the construction of section 2 with $G = \left(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta\right)^\wedge$, where Δ is a non-square integer prime to n. The order of G is $n\,\varphi_\Delta(n)$, so we set $k = n$ and $\lambda = \varphi_\Delta(n)$ and suppose that k and Λ are coprime.

Element of order n: As previously seen, we look for an element of order n in the kernel of the reduction modulo n from $\left(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta\right)^\wedge$ to $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$. This reduction is surjective by the Hensel Lemma. The element $g \equiv 1 + n\sqrt{\Delta} \pmod{n^2}$ is a generator of this kernel and g is indeed of order n as $g^r \equiv 1 + nr\sqrt{\Delta} \pmod{n^2}$ for all integer r. As a consequence of this expression of g^r, the discrete logarithm problem in $\langle g\rangle$ is easy.

k^{th} powers generation: To simplify, we suppose that Δ is neither a square modulo p nor modulo q. It is easy to see that the map $\alpha \mapsto \alpha/\overline{\alpha}$ from $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\times$ to $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ is surjective and that its kernel is $(\mathbf{Z}/n\mathbf{Z})^\times$. As a consequence, the map

$$\Psi : r \mapsto \frac{r+\sqrt{\Delta}}{r-\sqrt{\Delta}} = \frac{r^2+\Delta}{r^2-\Delta} + \frac{2r}{r^2-\Delta}\sqrt{\Delta},$$

from $\mathbf{Z}/n\mathbf{Z}$ to $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ is well-defined, injective and is almost surjective (we only miss 1 and elements that allow to factor n (elements different from 1 and that are congruent to 1 modulo p or 1 modulo q). Moreover, the map $\beta \mapsto \beta^n$ from $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$ to G^n is an isomorphism. As a consequence, the map

$$\mathbf{Z}/n\mathbf{Z} \to G^n : r \mapsto \Psi(r)^n,$$

is still injective and almost surjective.

Encryption function: The encryption function is

$$\begin{cases} \mathbf{Z}/n\mathbf{Z} \times (\mathbf{Z}/n\mathbf{Z})^{\times} \longrightarrow & G \\ (m\ ,\ r) & \longmapsto g^m \Psi(r)^n \end{cases}$$

where m is the plaintext and r a random element, and the public key is (n, Δ) where $n = pq$ is an RSA integer, Δ is a non-square integer, prime to n and Δ is neither a square modulo p nor modulo q.

Decryption algorithm: The trapdoor is $\lambda = \varphi_{\Delta}(n)$. The decryption algorithm is the same as the generic one. Note that it can be sped up by using Chinese remaindering (this is true for all the others schemes presented in this paper).

Security: The one-wayness of the scheme is based on the $\text{Class}_{G,k,g}$ problem and the reductions of Theorem 2 hold. The semantic security is based on the the difficulty of distinguishing the elements of G^n in G (As the map $r \mapsto \Psi(r)^n$ is injective and almost surjective, almost all the element of G^n can be produced. The ones that are not produced are either easy to distinguish or allow to factor n).

Expansion: The cryptosystem expansion is 4, *a priori*, but can be reduced to 3. One defines a lifting L of the elements of $\left(\mathcal{O}_{\Delta}/n\mathcal{O}_{\Delta}\right)^{\wedge}$ in $\left(\mathcal{O}_{\Delta}/n^2\mathcal{O}_{\Delta}\right)^{\wedge}$. Then, an element α of $\left(\mathcal{O}_{\Delta}/n^2\mathcal{O}_{\Delta}\right)^{\wedge}$ is represented by the couple $(k, \alpha \bmod n) \in \mathbf{Z}/n\mathbf{Z} \times \left(\mathcal{O}_{\Delta}/n\mathcal{O}_{\Delta}\right)^{\wedge}$ with k such that $\alpha = (1 + n\sqrt{\Delta})^k\, L(\alpha \bmod n)$. Note that the computation of this representation (by using the Hensel Lemmma) only costs a few multiplications and one inversion. This method can also be applied for the system of Galbraith.

Comparison with others additively homomorphic systems: In the following table, we compare this system with the Paillier and Galbraith schemes. The unity of complexity is the cost of a multiplication modulo n. We use the following estimations: a multiplication modulo n^2 costs as much as 3 multiplications modulo n (by using radix n representation), a multiplication modulo p^2 costs as much as a multiplication modulo n and three multiplications modulo p as much as a multiplication modulo n. An inversion modulo n costs as much as 10 multiplications modulo n. We have used Chinese remaindering for all the schemes.

Cryptosystem	Paillier	Galbraith	QF scheme
Group	$\left(\mathbf{Z}/n^2\mathbf{Z}\right)^{\times}$	$E/(\mathbf{Z}/n^2\mathbf{Z})$	$\left(\mathcal{O}_{\Delta}/n^2\mathcal{O}_{\Delta}\right)^{\wedge}$
Encryption	$\frac{9}{2}\left\|n\right\|_2 + 1$	$35\left\|n\right\|_2 + 3$	$9\left\|n\right\|_2 + 20$
Decryption	$\frac{3}{2}\left\|n\right\|_2 + \frac{5}{3}$	$21\left\|n\right\|_2 + \frac{5}{3}$	$3\left\|n\right\|_2 + \frac{4}{3}$

We see that the scheme in quadratic fields is much more faster than the system that uses elliptic curves, thanks to efficient exponentiation using Lucas

sequences. This scheme complexity is not far from the Paillier cryptosystem (the factor two is inherited from the respective costs of exponentiation in $\mathbf{Z}/n^2\mathbf{Z}$ and in $\left(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta\right)^\wedge$). As a result, this scheme is still practical.

If all the schemes are based on factorization, from Theorem 2, we see that the intermediate problems on which the one-wayness of the schemes are based are not the same. For Paillier, it is the RSA_n problem *i. e.*, the inversion of the map $x \mapsto x^n$ in $(\mathbf{Z}/n\mathbf{Z})^\times$. For the presented scheme, it is the adaptation of this problem in $(\mathcal{O}_\Delta/n\mathcal{O}_\Delta)^\wedge$, *i. e.*, the inversion of the map $\alpha \mapsto \alpha^n$. We do not know if one problem is easier than the other (as the only known way to solve them is to factor n), but this scheme brings some diversity as the Paillier scheme is almost the only practical additively homomorphic scheme known. Another advantage of this scheme is that one has more choice for the public key than for the Paillier scheme: one can choose freely the modulus n *and* the discriminant Δ.

Generalization: This scheme can also be generalized by working modulo n^{s+1} with $s > 1$ in order to encrypt messages of $\mathbf{Z}/n^s\mathbf{Z}$. One has only to find an element g of order n^s and an efficient algorithm for the discrete logarithm problem. One can see that the following element:

$$g := n\sqrt{\Delta} + 1 + \frac{1}{2}\Delta n^2 - \frac{1}{2^3}\Delta^2 n^4 + \frac{1}{2^4}\Delta^3 n^6 - \frac{5}{2^7}\Delta^4 n^8 + \cdots$$

obtained by successive applications of the Hensel Lemma is indeed of order n^s. Given g^k, one can still compute the discrete logarithm k at low cost, by computing recursively $k \mod n^2$, $k \mod n^4, \ldots$

IND-CCA2 *variant of this scheme:* Similarly to the Paillier cryptosystem, one can design a variant that is IND-CCA2 in the standard model. A cyclic group is obtained in the same way, by using primes p and q such that $(p - (\Delta/p))/2$ and $(q - (\Delta/q))/2$ are both primes. Then, one obtains a subgroup of order $n\,\varphi_\Delta(n)/2$. Note that some optimisations used by Cramer and Shoup in [CS02] to get compact ciphertexts for the adaptation of the Paillier scheme can also be done here as $\left(\mathcal{O}_\Delta/n^2\mathcal{O}_\Delta\right)^\wedge$ is very similar to $\left(\mathbf{Z}/n^2\mathbf{Z}\right)^\times$.

5 Conclusion

We have proposed two generic constructions that generalize many probabilistic cryptosystems already proposed. This process helps to capture the ideas behind these schemes. In particular, we have seen that the efficient homomorphic cryptosystem proposed in the group of norm 1 elements of a quadratic field is very similar to the Paillier scheme and can serve to construct an IND-CCA2 secure system in the standard model, which is a rare object. We hope that these generic constructions will help to propose new probabilistic cryptosystems. One possible domain of application could be class groups of quadratic orders such as those used in the NICE cryptosystem (cf. [PT00]).

References

[Ben88] Benaloh, J.C.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University (1988)

[BFP+01] Baudron, O., Fouque, P., Pointcheval, D., Poupard, G., Stern, J.: Practical multi-candidate election system. In: Proc. of PODC 2001 (2001)

[Cas07] Castagnos, G.: An efficient probabilistic public-key cryptosystem over quadratic fields quotients. Finite Fields Appl. 13(3), 563–576 (2007)

[CGH+01] Catalano, D., Gennaro, R., Howgrave-Graham, N., Nguyen, P.Q.: Paillier's cryptosystem revisited. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 206–214 (2001)

[CNS02] Catalano, D., Nguyen, P.Q., Stern, J.: The Hardness of Hensel Lifting: The Case of RSA and Discrete Logarithm. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 299–310. Springer, Heidelberg (2002)

[CS02] Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)

[Dem94] Demytko, N.: A New Elliptic Curve Based Analogue of RSA. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 40–49. Springer, Heidelberg (1994)

[DJ01] Damgård, I., Jurik, M.J.: A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992. Springer, Heidelberg (2001)

[FO99] Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)

[Gal02] Galbraith, S.D.: Elliptic Curve Paillier Schemes. Journal of Cryptology 15(2), 129–138 (2002)

[GM84] Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. 28, 270–299 (1984)

[GMMV03] Galindo, D., Martín, S., Morillo, P., Villar, J.L.: An Efficient Semantically Secure Elliptic Curve Cryptosystem Based on KMOV. In: Proc. of WCC 2003, pp. 213–221 (2003)

[Jur03] Jurik, M.: Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols. PhD thesis, Aarhus University (2003)

[KMOV92] Koyama, K., Maurer, U.M., Okamoto, T., Vanstone, S.A.: New Public-Key Schemes Based on Elliptic Curves over the Ring Z_n. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 252–266. Springer, Heidelberg (1992)

[Lip05] Lipmaa, H.: An Oblivious Transfer Protocol with Log-Squared Communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)

[NS98] Naccache, D., Stern, J.: A New Public Key Cryptosystem Based on Higher Residues. In: Proceedings of the Third ACM Conference on Computer and Communications Security, pp. 59–66 (1998)

[NSNK06] Nguyen, L., Safavi-Naini, R., Kurosawa, K.: Verifiable shuffles: a formal model and a Paillier-based three-round construction with provable security. Int. J. Inf. Secur. 5(4), 241–255 (2006)

[OU98] Okamoto, T., Uchiyama, S.: A New Public Key Cryptosystem as Secure as Factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)

[Pai99] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)

[PT00] Paulus, S., Takagi, T.: A new public-key cryptosystem over quadratic orders with quadratic decryption time. Journal of Cryptology 13, 263–272 (2000)

[SL93] Smith, P., Lennon, M.J.J.: LUC: A New Public Key System. In: Proc. of the Ninth IFIP Int. Symp. on Computer Security (1993), pp. 103–117 (1993)

Cramer-Shoup Satisfies a Stronger Plaintext Awareness under a Weaker Assumption

Isamu Teranishi[1] and Wakaha Ogata[2]

[1] NEC Corporation,
1753, Shimonumabe, Nakahara-Ku, Kawasaki, Kanagawa, 211-8666, Japan
teranisi@ah.jp.nec.com
[2] Tokyo Institute of Technology,
2-12-1 Ookayama, Meguro-ku Tokyo, 152-8550, Japan
wakaha@mot.titech.ac.jp

Abstract. In the seminal paper of Eurocrypt 2006, Dent defined a new assumption, *simulatability*, and showed that the well-known Cramer-Shoup public-key encryption scheme satisfied the weakest version of the plaintext awareness, the computational plaintext awareness, under the simulatability assumption, the DDH assumption, the DHK assumption, and the collision resistance of the hash function. However, a tricky aspect of the computational plaintext awareness was later shown. Moreover, the definition of the simulatability is elaborated. In this paper, we show that the Cramer-Shoup scheme satisfies a stronger variant of the plaintext awareness, the statistical plaintext awareness, under a weaker and simpler assumption than the simulatability. In particular, we show the *statistical* PA2-ness of the Cramer-Shoup scheme under *computational* assumptions.

Keywords: Statistical Plaintext Awareness, Standard Model, Cramer-Shoup Scheme.

1 Introduction

1.1 Background

Plaintext Awareness (PA2) [BR94,BDPR98,HLM03,BP04,D06,TO06,BD08] is one of the most fundamental notions about Public-Key Encryption schemes (PKEs). Intuitively, a PA2 secure PKE is a scheme such that an adversary cannot generate a ciphertext "without knowing" the corresponding plaintext. More precisely, a PKE is called PA2 secure, if there exists an extractor which can extract the plaintext from the ciphertext generated by the adversary.

Although PA2-ness was first defined in the random oracle model [BR94,BDPR98], Bellare and Palacio [BP04] succeeded in defining PA2-ness in the standard model. They gave three variants of standard model PA2-ness: perfect/statistical/computational PA2-ness, depending on whether the extracted plaintext was perfectly/statistically/computationally indistinguishable from the decryption. The PA2-nesses are important even in the

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 109–125, 2008.

standard model, because they can bring some insight into or an alternative perception of the design of existing PKEs, as said by Bellare and Palacio [BP04].

In the seminal paper [D06], Dent provided a sufficient condition for computational PA2-ness and showed the computational PA2-ness of the well-known Cramer-Shoup scheme [CS98] by using it.

Theorem 1 (Cramer-Shoup is Computationally PA2 Secure[D06]). The Cramer-Shoup scheme is computationally PA2 secure, if the underlying group satisfies the simulatability [D06], the DHK assumption [D91,BP04] and the DDH assumption, and also if the hash function is collision resistant.

Here, the *simulatability* is an assumption which Dent newly introduced in [D06]. The intuitive meaning of it is as follows: there exist polytime computable functions $\alpha : \{0,1\}^{\ell} \to \mathcal{G}$ and $\beta : \mathcal{G} \to \{0,1\}^{\ell}$ such that $\alpha \circ \beta$ and $\beta \circ \alpha$ are computationally indistinguishable from the identity maps. The intuitive meaning of the DHK assumption is as follows: "If an adversary $\mathsf{A}(g,h)$ outputs (u,v) such that (g,h,u,v) is a DDH-tuple, then A knows r satisfying $(u,v) = (g^r, h^r)$."

The above result is quite important for the study of the PA-ness, because the Cramer-Shoup scheme is the only known example of the standard model PA secure scheme. However, the formal definition of the underlying assumption, simulatability, is elaborate despite the simplicity of the intuition behind it. In fact, the formal definition allows a distinguisher to execute an "adaptively chosen message attack", and therefore prevents us from describing it simply.

Moreover, although Dent showed the computational PA2-ness of the Cramer-Shoup scheme, a tricky aspect of the computational PA2-ness was later shown:

Proposition 2 (Tricky Aspect of the Computationally PA2-ness[TO06]). *There exists a computationally PA2 secure PKE and an adversary such that no extractor can succeed in extracting the correct plaintext.*

It is also shown in [TO06] that there are no such PKE and an adversary in the case of the statistical PA-ness. Thus, we can say that the statistical PA2-ness is more similar to our intuition.

1.2 Our Results

Statistical PA2-ness of Cramer-Shoup Scheme: In this paper, we show that the Cramer-Shoup scheme satisfies a stronger PA2 security, statistical PA-ness, under assumption weaker and simpler than that of [D06]. That is, we introduce a weaker and simpler assumption, the *computationally random-like property* whose intuitive meaning is a "known message attack version of simulatability," and show the following theorem:

Theorem 3 (Cramer-Shoup is Statisitically PA2 Secure under Weaker Assumption). *The Cramer-Shoup scheme is statistically PA2 secure if the underlying group satisfies the computationally random-like property, the DDH assumption and the DHK assumption, and also if the hash function is collision resistant.*

We stress that we show the *statistical* PA2-ness of the Cramer-Shoup scheme under only *computational* assumptions. See Subsection 4.3, for the reason we can show the statistical PA-ness from the computational assumptions.

Sufficient Condition for the PA2-ness: In order to prove the PA2-ness of the Cramer-Shoup scheme, we give a sufficient condition for the statistical PA2-ness. Although Dent [D06] already gave another sufficient condition for the computational PA2-ness, our condition is not for the computational PA2-ness but for the statistical PA2-ness.

Moreover, our sufficient condition is formalized in a more practical way than that descrebed by Dent [D06]. That is, we formalize a part of our sufficient condition based on a slightly modified version of the IND-CCA2 game. Therefore, we can prove this part of our sufficient condition by slightly modifying the proof of the IND-CCA2 security of [CS98]. This means that we can prove the PA2-ness more easily.

2 Preliminary

In this section, we review the definition of notions described in Section 1.1. See Section 1.1 for the intuitions behind the definitions.

Definition 4 (Standard Model PA-ness[BP04]). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE. Let A, K, and P be polytime machines, which are respectively called *adversary*, *extractor*, and *plaintext creator*.

For plaintext creator P, its state st_P, and its random tape μ, we let $\mathsf{Enc}_{\mathsf{pk}} \circ \mathsf{P}(Q;\mu)$ denote the algorithm which executes the following procedures: $(M, \mathsf{st}_\mathsf{P}) \leftarrow \mathsf{P}(Q, \mathsf{st}_\mathsf{P}; \mu)$, $C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(M)$, and output C. Note that the state st_P was initialized to the null string ε.

For security parameter λ, we define two experiments $\mathsf{PA}^{\mathsf{Dec}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$ and $\mathsf{PA}^{\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$, shown in Fig. 1, where R_A, ρ, and μ are the random tapes of A, K, and P, List is the list of encryption queries of A, and st_K is the state of K.

We say that extractor K is *perfectly, statistically, or computationally successful* for A if, for any P, $\mathsf{PA}^{\mathsf{Dec}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$ and $\mathsf{PA}^{\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$ are perfectly, statistically, or computationally indistinguishable respectively.

We say that a PKE Π is *perfectly, statistically, or computationally PA2 secure* if for any adversary A, there exists a perfectly, statistically, or computationally successful extractor K for A. We also say that a PKE Π is *perfectly, statistically, or computationally PA1 secure* if, for any adversary A which makes no encryption query, there exists a perfectly, statistically, or computationally successful extractor K for A.

Definition 5 (DHK Assumption [D91,BP04]). Let λ be a security parameter. Let $\mathcal{G} = \mathcal{G}_\lambda$ be a family of cyclic groups with the prime order $q = q_\lambda$. Let A and K be polytime machines, named *adversary* and *extractor* respectively. We define an experiment $\mathsf{DHK}^{\mathsf{K}}_{\mathcal{G},\mathsf{A}}(\lambda)$ shown in Fig. 2. Here, NonDH is a symbol which

$\mathsf{PA}^{\mathsf{Dec}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$		$\mathsf{PA}^{\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.		$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
Take R_A and μ randomly.		Take R_A, ρ, and μ randomly.
	$*$	Initialize List and st_K to ε.
Run $\mathsf{A}(\mathsf{pk}; R_\mathsf{A})$.		Run $\mathsf{A}(\mathsf{pk}; R_\mathsf{A})$.
If A makes an encryption query (enc, Q)		If A makes an encryption query (enc, Q)
$C \leftarrow \mathsf{Enc}_\mathsf{pk} \circ \mathsf{P}(Q;\mu)$.	$*$	$C \leftarrow \mathsf{Enc}_\mathsf{pk} \circ \mathsf{P}(Q;\mu)$, $\mathsf{List} \leftarrow \mathsf{List}\|C$.
Send C to A as the reply.		Send C to A as the reply.
If A makes a decryption query (dec, Q)		If A makes a decryption query (dec, Q)
$M \leftarrow \mathsf{Dec}_\mathsf{sk}(Q)$.	$*$	$(M, \mathsf{st}_\mathsf{K}) \leftarrow \mathsf{K}(\mathsf{pk}, Q, R_\mathsf{A}, \mathsf{List}, \mathsf{st}_\mathsf{K}; \rho)$.
Send M to A as the reply.		Send M to A as the reply.
Return an output T of A.		Return an output T of A.

Fig. 1. Experiments for the Standard Model PA Security [BP04]

$\mathsf{DHK}^{\mathsf{K}}_{\mathcal{G},\mathsf{A}}(\lambda)$

Take random tapes R_A and ρ randomly.
$g \leftarrow \mathcal{G}$, $x \leftarrow \mathbb{Z}_q$, $h \leftarrow g^x$. Initialize st_K to ε.
Run $\mathsf{A}(g, h; R_\mathsf{A})$.
If A makes a query $Q = (u, v) \in \mathcal{G}^2$
$(r, \mathsf{st}_\mathsf{K}) \leftarrow \mathsf{K}(g, h, Q, R_\mathsf{A}, \mathsf{st}_\mathsf{K}; \rho)$.
If $r \in \mathbb{Z}_q$ and $(u, v) = (g^r, h^r)$, send r to A as the reply.
If $r = \mathsf{NonDH}$ and $v \neq u^x$, send r to A as the reply.
Otherwise, return 0 and terminate the experiment.
Return 1.

Fig. 2. Experiment for the DHK assumption

means that "I think that (g, h, u, v) is not a DH-tuple." We say that the *DHK assumption* on $\mathcal{G}$ holds, if $\mathcal{G}$ satisfies the following property:

$$\forall \mathsf{A} \exists \mathsf{K} \ : \ \Pr[\mathsf{DHK}^{\mathsf{K}}_{\mathcal{G},\mathsf{A}}(\lambda) \neq 1] \text{ is negligible for } \lambda.$$

Definition 6 (Simulatable Group [D06]). Let λ be a security parameter. Let $\mathcal{G} = \mathcal{G}_\lambda$ be a family of cyclic groups with the prime order $q = q_\lambda$. We say that $\mathcal{G}$ is *simulatable* if there exist a polynomial $\ell = \ell(\lambda)$, a deterministic polytime function $\alpha : \{0,1\}^\ell \to \mathcal{G}$ and a probabilistic polytime function $\beta : \mathcal{G} \to \{0,1\}^\ell$ satisfying the following properties:

(1) $\alpha(\beta(a;\rho)) = a$ holds for any $a \in \mathcal{G}$ and ρ.
(2) Let $\mathcal{O}^b_{\beta\circ\alpha}$ be the oracle such that, on inputting a symbol query, selects $R \in \{0,1\}^\ell$ and a random tape ρ of β randomly and outputs R or $\beta(\alpha(R);\rho)$, depending on whether $b = 0$ or $b = 1$. Then, for any polytime adversary A, the following probability is negligible:

$$|\Pr[b \leftarrow \{0,1\}, b' \leftarrow \mathsf{A}^{\mathcal{O}^b_{\beta\circ\alpha}}(1^\lambda) : b = b'] - \frac{1}{2}|$$

(3) Let $\mathcal{O}_{\alpha}^{b}$ be the oracle such that, on inputting a symbol query, selects $R \in \{0,1\}^{\ell}$ and $a \in \mathcal{G}$ randomly, outputs $\alpha(R)$ or a, depending on whether $b = 0$ or $b = 1$. Then, for any polytime adversary A, the following probability is negligible:

$$|\Pr[b \leftarrow \{0,1\}, b' \leftarrow \mathsf{A}^{\mathcal{O}_{\alpha}^{b}}(1^{\lambda}) : b = b'] - \frac{1}{2}|.$$

3 Sufficient Condition for Statistical PA2-ness

In [D06], Dent provided a very elegant idea for showing the PA2-ness, i.e. "if a ciphertext seems random, the encryption oracle is meaningless and therefore the PA2-ness is almost equivalent to the PA1-ness." Then he provided a sufficient condition for the computational PA2-ness by formalizing this idea.

In this section, we explain a new sufficient condition for the PA2-ness. Although our sufficient condition is also based on the Dent's above-mentioned idea. ours allows us to show not the computational PA2-ness but the statistical PA2-ness. Moreover, our condition is formalized in a more practical way, as described in Section 1.2.

3.1 Our Sufficient Condition

In order to formalize our sufficient condition, we introduce two notions, a *computationally random-like PKE* and an *EPA1 security*, whose intuitive meanings are "a ciphertext seems random" and "almost equivalent to the PA1-ness." Before giving them, we introduce a new notion, *computationally random-like set*, whose intuitive meaning is described in Section 1.

Definition 7 (Computationally Random-like Set). Let λ be a security parameter and $\mathcal{X} = \mathcal{X}_{\lambda}$ be a finite set parametrized by λ. We say that $\mathcal{X}$ is *computationally random-like* if there exists a polynomial $\ell = \ell(\lambda)$, a deterministic polytime function $\alpha : \{0,1\}^{\ell} \to \mathcal{X}$ and a probabilistic polytime function $\beta : \mathcal{X} \to \{0,1\}^{\ell}$ such that, for uniformly randomly selected $R \leftarrow \{0,1\}^{\ell}$, $a \leftarrow \mathcal{X}$, and a random tape ρ, the distributions of $(\alpha(R), R)$ and $(a, \beta(a;\rho))$ are computationally indistinguishable.

We now define the computationally random-like PKE. Intuitively, we say that a PKE is computationally random-like if there exists a computationally random-like set $\mathcal{X}$ such that a ciphertext is indistinguishable from a randomly selected element of $\mathcal{X}$, even if a distinguisher has access to a decryption oracle. The precise definition is as follows:

Definition 8 (Computationally Random-Like PKE). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE. We say that Π is *computationally random-like* if there exists a computationally random-like set $\mathcal{X} = \mathcal{X}_{\lambda}$ of (honestly or dishonestly generated) ciphertexts such that for any polytime adversary A,

$$|\Pr[b \leftarrow \{0,1\}, (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^{\lambda}), b' \leftarrow \mathsf{A}^{\mathsf{EoX}_{\mathsf{pk}}^{b}, \mathsf{Dec}_{\mathsf{sk}}}(\mathsf{pk}) : b = b'] - (1/2)|$$

is negligible.

$\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$		$\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A}}(\lambda)$
Take R_{A}, ρ, and μ randomly.		Take R_{A}, ρ, and μ randomly.
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{st}_{\mathsf{K}} \leftarrow \mathsf{List} \leftarrow \varepsilon$.		$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, $\mathsf{st}_{\mathsf{K}} \leftarrow \mathsf{List} \leftarrow \varepsilon$.
Run $\mathsf{A}(\mathsf{pk}; R_{\mathsf{A}})$.		Run $\mathsf{A}(\mathsf{pk}; R_{\mathsf{A}})$.
If A makes an encryption query (enc, Q),	*	If A makes an randomness query rand,
$C \leftarrow \mathsf{Enc}_{\mathsf{pk}} \circ \mathsf{P}(Q;\mu)$, $\mathsf{List} \leftarrow \mathsf{List}\|C$.	*	Select a bit string R randomly[1],
Send C to A as the reply.	*	Send R to A as the reply.
If A makes a decryption query (dec, Q),		If A makes a decryption query (dec, Q),
$M \leftarrow \mathsf{Dec}_{\mathsf{sk}}(Q)$.		$M \leftarrow \mathsf{Dec}_{\mathsf{sk}}(Q)$.
$(M', \mathsf{st}_{\mathsf{K}}) \leftarrow \mathsf{K}(\mathsf{pk}, Q, R_{\mathsf{A}}, \mathsf{List}, \mathsf{st}_{\mathsf{K}}; \rho)$.		$(M', \mathsf{st}_{\mathsf{K}}) \leftarrow \mathsf{K}(\mathsf{pk}, Q, R_{\mathsf{A}}, \mathsf{List}, \mathsf{st}_{\mathsf{K}}; \rho)$.
If $M \neq M'$,		If $M \neq M'$,
Return 0 and halt the experiment.		Return 0 and halt the experiment.
Send M to A as the reply.		Send M to A as the reply.
Return 1.		Return 1.

Fig. 3. Experiments for the Equality-PA2 (left) and for the Equality-PA1$^+$ (right)

Here, $\mathsf{EoX}^b_{\mathsf{pk}}$ is an oracle such that, if an adversary sends a plaintext M, it outputs $C_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(M)$ or $C_1 \leftarrow \mathcal{X}$, depending on whether $b = 0$ holds or not. A is not allowed to send to the decryption oracle answers from the $\mathsf{EoX}^b_{\mathsf{pk}}$-oracle.

We next introduce a variant of the PA security, *equality-PA (EPA)* security. Recall that the definition of PA-ness only requires that $M' \simeq \mathsf{Dec}_{\mathsf{sk}}(C)$ holds, where M' is an output of extractor, "$\simeq$" is indistinguishability, C is a decryption query of an adversary. Our EPA-ness is a variant of the PA-ness which requires not only that $M' \simeq \mathsf{Dec}_{\mathsf{sk}}(C)$ but $M' = \mathsf{Dec}_{\mathsf{sk}}(C)$ with overwhelming probability.

Definition 9 (Equality-PA Security). We take $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, A, K, and P as in Definition 4 and define $\mathsf{Enc}_{\mathsf{pk}} \circ \mathsf{P}(Q;\mu)$ as in Definition 4. For a security parameter λ, we define an experiment $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$ shown in the left of Fig. 3. We say that extractor K is *successful* for A, if it satisfies the following property:

$$\forall \mathsf{P} \ : \ \Pr[\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda) = 1] \text{ is negligible for } \lambda.$$

We let $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A}}(\lambda)$ denote $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A},\mathsf{Enc}\circ\mathsf{P}}(\lambda)$ if adversary A makes no encryption query to a plaintext creator P.

We say that PKE Π is *Equality-PA2 (EPA2) secure* if, for any A, there exists a successful extractor K. We say that PKE Π is *Equality-PA1 (EPA1) secure* if, for any polytime adversary A which makes no encryption query, there exists a successful extractor K.

Since equality implies the indistinguishability, the following theorem clearly holds:

Theorem 10 (EPA2 ⇒ Statistical PA2). *If a PKE is EPA2 secure, then it is statistically PA2 secure.*

[1] One can set the length of the bit string R arbitrarily, because A can obtain a bit string of arbitrary length by making randomness queries many times.

We now describe our sufficient condition. See Section 3.3 for the proof.

Theorem 11 (Comp. Random-Like + EPA1 ⇒ EPA2 (⇒ Stat. PA2)). *If a PKE is a computationally random-like and EPA1 secure, then it is EPA2 secure (and therefore is statistically PA2 secure).*

3.2 Randomness Oracle

Before showing Theorem 11, we review notions and a result given by Dent [D06]. In [D06], Dent defined an oracle named the *randomness oracle* which defined as follows: if an adversary makes a query a symbol query, it selects a random bit string R and sends R back to the adversary. Dent then gave a variant of the PA1 security, *PA1$^+$ security*, where an adversary was allowed to access the randomness oracle. He formalized his sufficient condition by using the PA1$^+$ security.

Since Dent had to carefully discuss about the difference from the PA1 security and the PA1$^+$ security, one may think that we also have to discuss carefully about the difference between the EPA1 security and the EPA1$^+$ security. Here the *EPA1$^+$ security* is a variant of the EPA1 security where an adversary was allowed to access the randomness oracle. However, such careful discussion is unnecessary because the EPA1$^+$ security is equivalent to EPA1 security.

In order to formalize the above discussion, we give the formal definition of EPA1$^+$ security. Let $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A}}(\lambda)$ be the experiment depicted at the right of Fig.3. We say that PKE Π is *EPA1$^+$ secure* if $\forall\mathsf{A}\exists\mathsf{K}\forall\mathsf{P}$: $\Pr[\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{K}}_{\Pi,\mathsf{A}}(\lambda) = 1]$ is negligible for λ.

Theorem 12 (EPA1 ⇔ EPA1$^+$). *A PKE is EPA1 secure if and only if it is EPA1$^+$ secure.*

Proof. Since EPA1$^+$ security clearly implies EPA1 security, we prove that the converse is also true. Let Π be an EPA1 secure PKE and A_0 be an adversary for the EPA1$^+$ security. We let R_{A_0} be the random tape of A_0, n_0 be the number of the randomness queries of A_0, and R_i be the answer to the i-th randomness query.

In order to show that A_0 has a successful extractor, we will construct adversary B_0 for the EPA1 security satisfying the following property: the behavior of $\mathsf{B}_0(\mathsf{pk}; R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{n_0})$ is the same as that of $\mathsf{A}_0(\mathsf{pk}; R_{\mathsf{A}_0})$ which is given R_i as the answer to the i-th randomness query. Here "$\mathsf{B}_0(\mathsf{pk}; R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{n_0})$" means that B_0 is given a public key pk as an input and $R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{n_0}$ as the random tape. Since the hypothesis ensures the existence of a successful extractor L_0 for B_0, we will construct the extractor K_0 for A_0 by using L_0.

We now describe the algorithm of adversary B_0 for the EPA security. For a public key pk and a random tape $\mathsf{R}_{\mathsf{B}_0}$, $\mathsf{B}_0(\mathsf{pk}; \mathsf{R}_{\mathsf{B}_0})$ parses its random tape $\mathsf{R}_{\mathsf{B}_0}$ as $R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{n_0}$ and executes $\mathsf{A}_0(\mathsf{pk}; \mathsf{R}_{\mathsf{A}_0})$. If A_0 makes the i-th randomness query, B_0 sends back R_i as an answer. If A_0 makes a decryption query, B_0 answers it by passing the queries to the decryption oracle. B_0 finally outputs the output of A_0.

From the assumption, there is a successful extractor, named L_0, of the EPA1$^+$ security for B_0. We construct an extractor K_0 for A_0 by using L_0. We denote how K_0 extracts the plaintext from C_{j_0}, where j_0 is an arbitrary integer and C_{j_0} is the j_0-th decryption query of A_0. Our basic idea behind the construction of K_0 is quite simple: "Since the behavior of B_0 is almost the same as that of A_0, L_0 cannot distinguish the j_0-th decryption query of B_0 from that of A_0. Thus, K_0 can extract the plaintext by feeding C_{j_0} to L_0."

We subtly modify the above idea, because A_0 and B_0 have one small but essential discrepancy. Recall that $\mathsf{B}_0(\mathsf{pk};\mathsf{B}_0) = \mathsf{B}_0(\mathsf{pk};R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{n_0})$ can be executed only if *all of* R_{A_0}, R_1, …, R_{n_0} *are given in advance* as the random tape. In contrast, when A_0 makes decryption query C_{j_0}, A_0 only knows R_{A_0}, R_1, …, $R_{k_{j_0}}$ but does not know $R_{k_{j_0}+1}$, …, R_{n_0}. Here k_{j_0} is the number of times that A_0 has made the randomness queries before it makes the j_0-th decryption query. From the definitions, extractor L_0 and K_0 have to extract plaintexts only from data which B_0 and A_0 know respectively. Thus, L_0 needs all of R_{A_0}, R_1, …, R_{n_0}, although K_0 can use only $R_1,\ldots,R_{k_{j_0}}$.

In order to resolve this discrepancy, K_0 selects $R'_{k_0+1},\ldots,R'_{n_0}$ randomly, sets $R^{[j_0]}_{\mathsf{B}_0} = R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{k_{j_0}}\|R'_{k_{j_0}+1}\|\cdots\|R'_{n_0}$, simulates $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$ with feeding $R^{[j_0]}_{\mathsf{B}_0}$ as the random tape of B_0, obtains an answer of L_0 to the j_0-th decryption query of B_0, and outputs the answer.

We now give the precise description of K_0. Since K_0 uses the same algorithm as A_0 as a subroutine of B_0, we denote the subroutine as $\overline{\mathsf{A}}_0$ in order to distinguish the subroutine from A_0 itself. The inputs of K_0 are a public key pk, the ciphertext C_{j_0}, the random tape R_{A_0} of A_0, the list $\mathsf{List} = R_1\|\cdots\|R_{k_{j_0}}$ of the answers from the randomness oracle, the state $\mathsf{st}_{\mathsf{K}_0}$, and the random tape ρ_{K_0} of K_0. On inputting them, K_0 parses ρ_{K_0} as $R'_{k_{j_0}+1}\|\ldots\|R'_{n_0}\|\rho_{\mathsf{L}_0}$, sets $R^{[j_0]}_{\mathsf{B}_0} = R_{\mathsf{A}_0}\|R_1\|\cdots\|R_{k_{j_0}}\|R'_{k_{j_0}+1}\|\cdots\|R'_{n_0}$, initializes $\mathsf{st}_{\mathsf{L}_0}$ to ε and executes $\overline{\mathsf{A}}_0(\mathsf{pk};R_{\mathsf{A}_0})$. If $\overline{\mathsf{A}}_0$ makes the i-th query to the randomness oracle for $i \le k_{j_0}$, K_0 sends R_i back to $\overline{\mathsf{A}}_0$. If $\overline{\mathsf{A}}_0$ makes the i-th query to the randomness oracle for $i > k_{j_0}$, the extraction has failed. In this case, K_0 outputs fail and terminates, (although we can prove that the extraction never failed). If $\overline{\mathsf{A}}_0$ makes the j-th decryption query $\overline{C}_j$ for $j < j_0$, K_0 computes $(M^{[j_0]}_j,\mathsf{st}_{\mathsf{L}_0}) \leftarrow \mathsf{L}_0(\mathsf{pk},\overline{C}_j,R^{[j_0]}_{\mathsf{B}_0},\mathsf{st}_{\mathsf{L}_0};\rho_{\mathsf{L}_0})$ and sends $M^{[j_0]}_j$ back to $\overline{\mathsf{A}}_0$. If $\overline{\mathsf{A}}_0$ makes the j_0-th decryption query $\overline{C}_{j_0}$, K_0 computes $(M^{[j_0]}_{j_0},\mathsf{st}_{\mathsf{L}_0}) \leftarrow \mathsf{L}_0(\mathsf{pk},\overline{C}_{j_0},R^{[j_0]}_{\mathsf{B}_0},\mathsf{st}_{\mathsf{L}_0};\rho_{\mathsf{L}_0})$, outputs $(M^{[j_0]}_{j_0},\mathsf{st}_{\mathsf{K}_0})$, and terminates.

The only difference between the behavior of K_0 and that of $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$ is as follows: K_0 does not check whether $M^{[j_0]}_j = \mathsf{Dec}_{\mathsf{sk}}(\overline{C}_j)$ holds or not, although $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$ does. However, since the output $M^{[j_0]}_j$ of extractor L_0 for the EPA1$^+$ security is equal to $\mathsf{Dec}_{\mathsf{sk}}(\overline{C}_j)$ with overwhelming probability, this checking is unnecessary. Therefore, the messages $M^{[j_0]}_1,\ldots,M^{[j_0]}_{j_0}$ generated by K_0 are equal to $\mathsf{Dec}_{\mathsf{sk}}(\overline{C}_1),\ldots,\mathsf{Dec}_{\mathsf{sk}}(\overline{C}_{j_0})$ with overwhelming probability.

By using induction, we show that the output $M_{j_0}^{[j_0]} = \mathsf{Dec}_{\mathsf{sk}}(\overline{C}_{j_0})$ of K_0 is equal to $\mathsf{Dec}_{\mathsf{sk}}(C_{j_0})$ with overwhelming probability. Suppose that K_0 succeeds in outputting the decrypted plaintexts for $j < j_0$. From the definition of K_0, adversary A_0 and the subroutine $\overline{\mathsf{A}}_0$ of K_0 are given the same input pk, the same random tape R_{A_0}, and the same answer $R_1, \ldots, R_{k_{j_0}}$ to the randomness queries. From the induction hypothesis, both A_0 and $\overline{\mathsf{A}}_0$ are given the decrypted plaintexts as an answers for the the first, ..., $(j_0 - 1)$-th decryption queries. These facts mean that A_0 and $\overline{\mathsf{A}}_0$ makes the same j_0-th decryption query. That is, $C_{j_0} = \overline{C}_{j_0}$ holds. Hence, $M_{j_0}^{[j_0]} = \mathsf{Dec}_{\mathsf{sk}}(\overline{C}_{j_0}) = \mathsf{Dec}_{\mathsf{sk}}(C_{j_0})$. This means that K_0 is successful. □

We can see that the above proof becomes invalid if we consider not the EPA security but computational PA security. Recall that the computational PA-ness ensures the computational indistinguishability between a tuple $(M_1, \ldots, M_n)$ of outputs of extractor and $(\mathsf{Dec}_{\mathsf{sk}}(\overline{C}_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(\overline{C}_n))$, *only if all* $\overline{C}_i$ *are output by the same adversary* B_0 *with the same random tape* R_{B_0}. Here $\mathsf{Dec}_{\mathsf{sk}}(\overline{C}_i)$ is the i-th decryption query of B_0.

In the above proof, K_0 inputs new $R_{\mathsf{B}_0}^{[j_0]}$ to extractor L_0 each times K_0 is executed. Therefore, we cannot conclude that the tuple $(M_1^{[1]}, \ldots, M_{n_0}^{[n_0]})$ of outputs of L_0 is computationally indistinguishable from $(\mathsf{Dec}_{\mathsf{sk}}(C_1), \ldots, \mathsf{Dec}_{\mathsf{sk}}(C_{n_0}))$, if we only assume the computational PA-security of Π. Since an answer of K_0 to the j_0-th encryption query of A_0 is $M_{j_0}^{[j_0]}$, this means that K_0 may not be a successful extractor for the computational PA2 security.

3.3 Proof of Theorem 11

Proof. Let Π be a PKE which is computationally random-like and EPA1 secure. From Theorem 12, Π is EPA1$^+$ secure. Since Π is computationally random-like, there exist a computationally random-like set $\mathcal{X} = \mathcal{X}_\lambda$ and functions $\alpha : \{0,1\}^\ell \to \mathcal{X}$ and $\beta : \mathcal{X} \to \{0,1\}^\ell$ satisfying the property described in Definition 8 and 7.

Let A_0 be an adversary for the EPA2 security of Π and n_0 be the number of steps of A_0. In order to show that A_0 has a successful extractor, we will construct an adversary B_0 for the EPA1$^+$ security, which executes A_0, obtains random bits $R'_1, \ldots, R'_{n_0}$ from the randomness oracle, and feeds $C'_1 = \alpha(R'_1), \ldots, C'_n = \alpha(R'_n)$ to A_0 as an answer to the encryption queries. The EPA1$^+$ property of Π ensures the existence of extractor L_0 for B_0. We will construct extractor K_0 for A_0 by using L_0.

The precise description of adversary B_0 for the EPA1$^+$ security is as follows. $\mathsf{B}_0(\mathsf{pk}; R)$ executes $\mathsf{A}_0(\mathsf{pk}; R)$. If A_0 makes a decryption query, B_0 answers it by making a decryption query. If A_0 makes the i-th encryption query Q_i, B_0 sends query to the randomness oracle, receives an answer R'_i, computes $C'_i = \alpha(R'_i)$, and sends C'_i back to A_0. Finally, B_0 outputs the output of A_0.

From the EPA1$^+$ security of Π, there exists an extractor L_0 for B_0. We construct extractor K_0 for A_0 by using L_0. We would like to denote how K_0 extracts

the plaintext from $\hat{C}_j$, where j is an arbitrary integer and $\hat{C}_j$ is the j-th decryption query of A_0. Let k be the number of times that A_0 has made the encryption queries before it makes the j-th decryption query.

Our idea behind the construction of K_0 is as follows: K_0 obtains $\mathsf{Dec}_{\mathsf{sk}}(\hat{C}_j)$ by feeding $\hat{C}_j$ to L_0. However, recall that L_0 is an extractor for adversary B_0 of the EPA1$^+$ security although K_0 is an extractor for adversary A_0 of the statistical PA2 security. This means that L_0 requires the list $\mathsf{List}' = R'_1\|\cdots\|R'_k$ as an input although K_0 is given the list $\mathsf{CList} = C_1\|\cdots\|C_k$ as an input. Here List' is the list of the answers from the randomness oracle to B_0 and CList is the list of the answers from the encryption oracle to A_0. Therefore, K_0 selects $\rho_1, \ldots, \rho_k$ randomly, sets $R_1 = \beta(C_1; \rho_1)$, …, $R_k = \beta(C_k; \rho_k)$, and feeds not List' but $\mathsf{List} = R_1\|\cdots\|R_k$ to L_0.

The precise description of K_0 is as follows. $\mathsf{K}_0(\mathsf{pk}, \hat{C}_j, R, \mathsf{CList}, \mathsf{st}; \rho_{\mathsf{K}_0})$ parses CList as $\mathsf{CList} = C_1\|\cdots\|C_k$, parses ρ_{K_0} as $\rho_{\mathsf{K}_0} = \rho_{\mathsf{L}_0}\|\rho_1\|\cdots\|\rho_{n_0}$, computes $R_1 = \beta(C_1; \rho_1)$, …, $R_k = \beta(C_k; \rho_k)$, sets $\mathsf{List} = R_1\|\cdots\|R_k$, obtains $(\hat{M}_j, \mathsf{st}) \leftarrow \mathsf{L}_0(\mathsf{pk}, \hat{C}_j, R, \mathsf{List}, \mathsf{st}; \rho_{\mathsf{L}_0})$, and outputs $(\hat{M}_j, \mathsf{st})$.

Finally we show that extractor K_0 for A_0 is successful. We fix an arbitrary plaintext creator P_0. Our idea behind the proof of the successfulness of K_0 is as follows. Since L_0 is successful, output $\hat{M}_j$ of $\mathsf{L}_0(\cdots, \hat{C}_j, \cdots, \mathsf{List}', \cdots)$ is equal to $\mathsf{Dec}_{\mathsf{sk}}(\hat{C}_j)$ with overwhelming probability in $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$. From the definitions of $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$ and B_0, the experimenter of $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$ feeds to L_0 the list List' of answers $R'_1, \ldots, R'_k$ from the randomness oracle, and B_0 feeds $C'_1 = \alpha(R'_1)$, …, $C'_k = \alpha(R'_k)$ to its subroutine A_0. Since $\mathcal{X}$ is random-like, $(C'_i, R'_i) = (\alpha(R'_i), R'_i)$ is computationally indistinguishable from $(C''_i, \beta(C''_i, \rho_i))$, where C''_i is a randomly selected element of $\mathcal{X}$ and ρ_i is a randomly selected bit string. Since Π is random-like, a randomly selected element C''_i of $\mathcal{X}$ is computationally indistinguishable from a ciphertext $C_i = \mathsf{Enc}_{\mathsf{pk}}(M_i; r_i)$. Here M_i is a plaintext computed by a plaintext creator P_0 from the i-th encryption query of A_0, and r_i is a random tape. In particular, if we set $R''_i = \beta(C''_i, \rho_i)$ and $R_i = \beta(C_i, \rho_i)$, R''_i is computationally indistinguishable from R_i.

Therefore, output $\hat{M}_j$ of $\mathsf{L}_0(\cdots, \hat{C}_j, \cdots, \mathsf{List}, \cdots)$ is equal to $\mathsf{Dec}_{\mathsf{sk}}(\hat{C}_j)$ with overwhelming probability, even in the following experiment: A_0 is fed $C_1 = \mathsf{Enc}_{\mathsf{pk}}(M_1; r_1)$, …, $C_k = \mathsf{Enc}_{\mathsf{pk}}(M_k; r_k)$ as answers to encryption queries, and L_0 is fed $\mathsf{List} = R_1\|\cdots\|R_k = \beta(C_1; \rho_1)\|\cdots\|\beta(C_k; \rho_k)$. Since K_0 feeds $\mathsf{List} = \beta(C_1; \rho_1)\|\cdots\|\beta(C_k; \rho_k)$ to L_0, the above experiment is the same as $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}_0}_{\Pi,\mathsf{A}_0}(\lambda)$. Thus, output $\hat{M}_j$ of $\mathsf{K}_0(\cdots, \hat{C}_j, \cdots, \mathsf{Clist}, \cdots)$ is equal to $\mathsf{Dec}_{\mathsf{sk}}(\hat{C}_j)$ with overwhelming probability, where $\mathsf{Clist} = C_1\|\cdots\|C_k = \mathsf{Enc}_{\mathsf{pk}}(M_1; r_1)\|\cdots\|\mathsf{Enc}_{\mathsf{pk}}(M_k; r_k)$. This means that K_0 is successful.

We now formally prove that K_0 is successful. We fix an arbitrary plaintext creator P_0 and construct a distinguisher D_0 for the random-like property of $\mathcal{X}$ and an adversary C_0 for the game (depicted in Definition 8) of the random-like property of Π. Let (C^*, R^*) be an instance of for distinguishing game of Definition 7. D_0 would like to know whether $C^* = \alpha(R^*)$ holds or $R^* = \beta(C^*; \rho^*)$ holds for some random tape ρ^*. $\mathsf{D}_0(C^*, R^*)$ obtains $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, selects

$i_0 \in \{1, \ldots, n_0\}$ randomly, selects R and ρ_{L_0} randomly, initializes $\mathsf{List}^\dagger$ and st to ε, and executes $\mathsf{A}_0(\mathsf{pk}; R)$. If A_0 makes the i-th encryption query Q_i for $i < i_0$, D_0 selects $C_i'' \in \mathcal{X}$ and ρ_i randomly, computes $R_i'' = \beta(C_i''; \rho_i)$, resets $\mathsf{List}^\dagger \leftarrow \mathsf{List}^\dagger \| R_i''$, and sends C_i'' back to A_0. If A_0 makes the i-th encryption query Q_{i_0}, D_0 resets $\mathsf{List}^\dagger \leftarrow \mathsf{List}^\dagger \| R^*$, and sends C^* back to A_0. If A_0 makes the i-th encryption query Q_i for $i > i_0$, D_0 selects $R_i' \in \{0,1\}^\ell$ randomly, computes $C_i' = \alpha(R_i')$, resets $\mathsf{List}^\dagger \leftarrow \mathsf{List}^\dagger \| R_i'$, and sends C_i' back to A_0. If A_0 makes the j-th decryption query $\hat{C}_j$, D_0 computes $\hat{M}_j \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\hat{C}_j)$ and $(\hat{M}_j', \mathsf{st}) \leftarrow \mathsf{L}_0(\mathsf{pk}, \hat{C}_j, R, \mathsf{List}^\dagger, \mathsf{st}; \rho_{\mathsf{L}_0})$. If $\hat{M}_j \neq \hat{M}_j'$ holds, D_0 outputs 0 and terminates. Otherwise, D_0 sends $\hat{M}_j$ back to A_0. If A_0 terminates, D_0 outputs 1 and terminates.

We next construct an adversary C_0 for the game described in Definition 8. Let pk_* be an instance of this game. $\mathsf{C}_0(\mathsf{pk}_*)$ sets $\mathsf{pk} = \mathsf{pk}_*$, and selects $i_0 \in \{1, \ldots, n_0\}$ randomly, selects random tapes R, ρ_{L_0}, and μ randomly, initializes st, $\mathsf{st}_{\mathsf{P}_0}$, and $\mathsf{List}^\ddagger$ to ε, and executes $\mathsf{A}_0(\mathsf{pk}; R)$. If A_0 makes the i-th encryption query Q_i for $i < i_0$, C_0 selects a random tape r_i randomly, computes $(M_i, \mathsf{st}_{\mathsf{P}_0}) \leftarrow \mathsf{P}_0(Q_i, \mathsf{st}_{\mathsf{P}_0}; \mu)$ and $C_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(M_i; r_i)$, computes $R_i \leftarrow \beta(C_i; \rho_i)$, resets $\mathsf{List}^\ddagger \leftarrow \mathsf{List}^\ddagger \| R_i$, and sends C_i back to A_0. If A_0 makes the i_0-th encryption query Q_{i_0}, C_0 computes $(M_{i_0}, \mathsf{st}_{\mathsf{P}_0}) \leftarrow \mathsf{P}_0(Q_{i_0}, \mathsf{st}_{\mathsf{P}_0}; \mu)$, makes query M_{i_0} to EoX-oracle, obtains an answer $C_{i_0}^*$, computes $R_{i_0}^* \leftarrow \beta(C_{i_0}^*; \rho_{i_0})$, resets $\mathsf{List}^\ddagger \leftarrow \mathsf{List}^\ddagger \| R_{i_0}^*$, and sends $C_{i_0}^*$ back to A_0. If A_0 makes the i-th encryption query Q_i for $i < i_0$, C_0 selects $C_i'' \in \mathcal{X}$ and ρ_i randomly, computes $R_i'' \leftarrow \beta(C_i''; \rho_i)$, resets $\mathsf{List}^\ddagger \leftarrow \mathsf{List}^\ddagger \| R_i''$, and sends C_i'' back to A_0. If A_0 makes the j-th decryption query $\hat{C}_j$, C_0 makes decryption query $\hat{C}_j$, obtains an answer $\hat{M}_j$, and computes $(\hat{M}_j', \mathsf{st}) \leftarrow \mathsf{L}_0(\mathsf{pk}, \hat{C}_j, R, \mathsf{List}^\ddagger, \mathsf{st}; \rho_{\mathsf{L}_0})$. If $\hat{M}_j \neq \hat{M}_j'$ holds, C_0 outputs 0 and terminates. Otherwise, C_0 sends $\hat{M}_j$ back to A_0. If A_0 terminates, C_0 outputs 1 and terminates.

One can easily show that the following the distributions of 1. and 2. are equal, the distributions of 3. and 4. are equal, and the distributions of 5. and 6. are equal.

1. An output of $\mathsf{EPA1}^{+\mathsf{Dec},\mathsf{L}_0}_{\Pi,\mathsf{B}_0}(\lambda)$.
2. An output of D_0, in the case where $i_0 = 1$ and $C_* = \alpha(R_*)$ hold.
3. An output of D_0, in the case where $i_0 = n_0$ and $R_* = \beta(C_*; \rho_*)$ hold.
4. An output of C_0, in the case where $i_0 = 1$ hold and C_{i_0} is a randomly selected element of $\mathcal{X}$.
5. An output of C_0, in the case where $i_0 = n_0$ hold and $C_{i_0} = \mathsf{Enc}_{\mathsf{pk}}(M_{i_0})$ holds.
6. An output of $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}_0}_{\Pi,\mathsf{A}_0}(\lambda)$.

Since PKE Π is computationally random-like, a hybrid argument shows that the distribution of 2. and 3. are computationally indistinguishable and the distribution of 4. and 5. are computationally indistinguishable also. Therefore, the theorem holds. □

4 Statistical PA2-ness of Cramer-Shoup Scheme

In this section, we show that the Cramer-Shoup scheme is statistically PA2 secure under a weaker assumption than Dent assumed in [D06]. The description of the Cramer-Shoup scheme is reviewed in Fig.4.

4.1 Our Assumption Is Weaker Than Dent's One[D06]

First, we show that our assumption, the random-like property, is weaker than Dent's one [D06], simulatability.

Theorem 13 (Simulatable Group ⇒ Random-Like Group). *If a prime order cyclic group $\mathcal{G}$ is simulatable, then it is computationally random-like.*

Proof. Suppose that there exists a group $\mathcal{G}$ which is not computationally random-like but is simulatable. From the simulatability of $\mathcal{G}$, there exists a polynomial $\ell = \ell(\lambda)$, a deterministic polytime function $\alpha : \{0,1\}^{\ell} \to \mathcal{G}$ and a probabilistic polytime function $\beta : \mathcal{G} \to \{0,1\}^{\ell}$ satisfying the properties (1), (2), and (3) of Definition 6.

We show that (ℓ, α, β) satisfies the condition of the computationally random-like property of $\mathcal{G}$. That is, we show that, for any polytime distinguisher D, $|\Pr[\mathsf{D}(\alpha(R), R) = 1] - \Pr[\mathsf{D}(a, \beta(a;\rho)) = 1]|$ is negligible. Here $R \in \{0,1\}^{\ell}$, ρ, and $a \in \mathcal{G}$ are randomly selected.

To this end, we take an arbitrary distinguisher D_0. By using D_0, we construct polytime adversaries B_2 and B_3 for the experiments of (2) and (3) of Definition 6. $\mathsf{B}_2(1^{\lambda})$ sends query to $\mathcal{O}^b_{\beta\circ\alpha}$, obtains R_* as an answer, executes $\mathsf{D}_0(\alpha(R_*), R_*)$, obtains an outputs b' of D_0, and outputs b'. In contrast, $\mathsf{B}_3(1^{\lambda})$ sends query to $\mathcal{O}^b_{\alpha}$, obtains a_* as an answer, selects ρ randomly, executes $\mathsf{D}_0(a_*, \beta(a_*;\rho))$, obtains an outputs b' of D_0, and outputs b'.

One can easily show that the following properties hold:

- The distribution of $\mathsf{D}_0(\alpha(R_*), R_*)$ for randomly selected $R_* \in \{0,1\}^{\ell}$ is the same as that of an output of $\mathsf{B}_2{}^{\mathcal{O}^0_{\beta\circ\alpha}}$.
- The distribution of an output of $\mathsf{B}_2{}^{\mathcal{O}^1_{\beta\circ\alpha}}$ is the same as that of an output of $\mathsf{B}_3{}^{\mathcal{O}^0_{\alpha}}$, because the property (1) of Definition 6 implies $(\alpha(\beta(\alpha(R_*);\rho)), \beta(\alpha(R_*);\rho)) = (\alpha(R_*), \beta(\alpha(R_*);\rho))$.
- The distribution of an output of $\mathsf{B}_3{}^{\mathcal{O}^1_{\alpha}}$ is the same as that of $\mathsf{D}_0(a, \beta(a))$ for randomly selected $a \in \mathcal{G}$.

—$\mathsf{Gen}(1^{\lambda})$—	—$\mathsf{Enc}_{\mathsf{pk}}(M)$—	—$\mathsf{Dec}_{\mathsf{sk}}(C)$—
$g, h \leftarrow \mathcal{G}$, $z, z', x, x', y, y' \leftarrow \mathbb{Z}_q$.	$r \leftarrow \mathbb{Z}_q$	$\theta \leftarrow H(u, v, e)$.
$(b, c, d) \leftarrow (g^z h^{z'}, g^x h^{x'}, g^y h^{y'})$.	$(u, v, e) \leftarrow (g^r, h^r, Mb^r)$.	If $u^{x+\theta y} v^{x'+\theta y'} \neq \pi$,
$\mathsf{pk} \leftarrow (g, h, b, c, d)$.	$\theta \leftarrow H(u, v, e)$.	output $\perp$.
$\mathsf{sk} \leftarrow (z, z', x, x', y, y')$.	$\pi \leftarrow (cd^{\theta})^r$.	Otherwise,
Output $(\mathsf{pk}, \mathsf{sk})$.	Output $C = (u, v, e, \pi)$.	output $e/u^z v^{z'}$.

Fig. 4. The Cramer-Shoup Scheme

Since $|\Pr[\mathsf{D}_0(\alpha(R_*), R_*) = 1] - \Pr[\mathsf{D}_0(a_*, \alpha(a_*; \rho_*)) = 1]|$ is non-negligible, this means that the winning probability of B_2 or B_3 is non-negligible. This contradicts the property (2) or (3) of Definition 6.

4.2 Proof of Main Theorem

We finally show our main theorem, Theorem 3. From our sufficient condition (Theorem 11), all we have to prove is that the Cramer-Shoup scheme is computationally random-like and is EPA1 secure. Recall that our experiment for the definition of a random-like PKE is quite similar to that for the definition of the IND-CCA2 security. Therefore, we can prove this part of our sufficient condition by slightly modifying the proof of the IND-CCA2 security of [CS98].

Proof (The Cramer-Shoup Scheme is Random-Like). We set $\mathcal{X} = \mathcal{G}^4$. Since $\mathcal{G}$ is computationally random-like, $\mathcal{G}^4$ is clearly computationally random-like. Recall the proof [CS98] of the IND-CCA2 security of the Cramer-Shoup scheme. In the proof, the authors of [CS98] showed that a ciphertext is indistinguishable from a random element of $\mathcal{G}^4$ even if an adversary can access the decryption oracle. This means that the winning advantage that an adversary enjoys the game descrebed in Definition 8 is negligible. □

We next prove the EPA1 security of the Cramer-Shoup scheme. To do so, we introduce a new notion, *regularity*. Intuitively, we say that a PKE is *regular* if, for any ciphertext C satisfying $\mathsf{Dec}_{\mathsf{sk}}(C) \neq \perp$, there exists (M, r) satisfying $C = \mathsf{Enc}_{\mathsf{pk}}(M; r)$ with overwhelming probability. This property clearly holds if C is an honestly generated ciphertext. The essence of the regularity is that the property holds even if C is maliciously generated. The precise definition of the regularity is as follows:

Definition 14 (Regularity). Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. For a public key/secret key pair $(\mathsf{pk}, \mathsf{sk})$ of Π, let $\mathcal{D}_{\mathsf{sk}}$ be the set of the bit string C satisfying $\mathsf{Dec}_{\mathsf{sk}}(C) \neq \perp$, and let $\mathcal{E}_{\mathsf{pk}}$ be the set of the bit string C satisfying $C = \mathsf{Enc}_{\mathsf{pk}}(M; r)$ for some M and r.

We say that Π is *regular*, if for any pk_0 and C_0,

$$\Pr[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), C_0 \in \mathcal{D}_{\mathsf{sk}} \setminus \mathcal{E}_{\mathsf{pk}_0} \mid \mathsf{pk} = \mathsf{pk}_0]$$

is negligible for λ.

Note that some artificial PKEs do not satisfy the regularity. See the full paper for an example. One can easily show the following lemma:

Lemma 15. *The Cramer-Shoup scheme is regular.*

We now prove EPA1 security of the Cramer-Shoup scheme.

Proof (The Cramer-Shoup Scheme is EPA1 Secure). Since the underlying group $\mathcal{G}$ is computationally random-like, there exists a polynomial $\ell = \ell(\lambda)$, and

functions $\alpha : \{0,1\}^\ell \to \mathcal{G}$ and $\beta : \mathcal{G} \to \{0,1\}^\ell$ satisfying the property described in Definition 7.

Let A_0 be an adversary against EPA1 security and n_0 be the number of steps of A_0. We have to construct an extractor for A_0. A basic strategy for constructing an extractor for A_0 is as follows. Let $\mathsf{pk} = (g, h, b, c, d)$ be a public key. We construct adversary B_0 for the DHK property, which executes A_0, obtains decryption queries $C_1 = (u_1, v_1, e_1, \pi_1), C_2 = (u_2, v_2, e_2, \pi_2), \ldots$ of A_0, and makes a query (u_i, v_i) for each i. Then the DHK assumption ensures the existence of extractor L_0 for B_0. From the definition, L_0 succeeds in outputting r_i with overwhelming probability. Here r_i is an element of $\mathbb{Z}_q$ satisfying $(u_i, v_i) = (g^{r_i}, h^{r_i})$. (To simplify, we here omit to consider the case where $r_i = \mathsf{NonDH}$.) The plaintext $M_i = \mathsf{Dec}_{\mathsf{sk}}(C_i)$ is easily computable from C_i, if we know the random tape r_i of $C_i = \mathsf{Enc}_{\mathsf{pk}}(M_i; r_i) = (u_i, v_i, e_i, \pi_i) = (g^{r_i}, h^{r_i}, M_i b^{r_i}, (cd^{\theta_i})^{r_i})$. Here $\theta_i = H(u_i, v_i, e_i)$. Therefore, we can construct extractor K_0 for A_0 by using L_0.

However, we have to subtly modify the above basic strategy, because there is a small discrepancy between B_0 and A_0. Let R_{A_0} and R_{B_0} be the random tapes of A_0 and B_0. From the definition of the DHK assumption, B_0 are given g, h and R_{B_0} only, although A_0 are given g, h, b, c, d and R_{A_0}. Thus, in order to execute A_0, B_0 has to construct $(b, c, d, R_{\mathsf{A}_0})$ *in a deterministic way*, by using its input (g, h, R_{B_0}) only. Therefore, B_0 parses R_{B_0} as $R_{\mathsf{B}_0} = R_b \| R_c \| R_d \| R_{\mathsf{A}_0}$, and sets $(b, c, d) = (\alpha(R_b), \alpha(R_c), \alpha(R_d))$.

The precise description of B_0 is as follows. $\mathsf{B}_0(g, h; R_{\mathsf{B}_0})$ parses R_{B_0} as $R_{\mathsf{B}_0} = R_b \| R_c \| R_d \| R_{\mathsf{A}_0}$, computes $(b, c, d) = (\alpha(R_b), \alpha(R_c), \alpha(R_d))$, sets $\mathsf{pk} = (g, h, b, c, d)$, and executes $\mathsf{A}_0(\mathsf{pk}; R_{\mathsf{A}_0})$. If A_0 makes the i-th decryption query $C_i = (u_i, v_i, e_i, \pi_i)$, B_0 makes query (u_i, v_i) and obtains answer r_i. If $r_i = \mathsf{NonDH}$ holds, B_0 sends $\perp$ to A_0. Otherwise, B_0 computes $\theta_i = H(u_i, v_i, e_i)$ and $\pi_i' = (cd^{\theta_i})^{r_i}$. If $\pi_i = \pi_i'$ holds, B_0 sends e_i / b^{r_i} to A_0. Otherwise, B_0 sends $\perp$ to A_0. If A_0 terminates, B_0 terminates.

From the DHK assumption, there exists an extractor L_0 of the DHK property for B_0. By using L_0 as a subroutine, we construct an extractor K_0 of A_0 for the EPA1 property. The basic strategy to construct K_0 has already been described. However, we have to modify the basic strategy because of the previously mentioned discrepancy between B_0 and A_0. From the definitions, extractor L_0 and K_0 have to extract plaintexts only from data known by B_0 and A_0 respectively. Recall that B_0 is given $(g, h, R_{\mathsf{B}_0}) = (g, h, R_b \| R_c \| R_d \| R_{\mathsf{A}_0})$, although A_0 is given $(g, h, b, c, d, R_{\mathsf{A}_0}) = (g, h, \alpha(R_b), \alpha(R_c), \alpha(R_d), R_{\mathsf{A}_0})$. That is, B_0 knows (R_b, R_c, R_d) although A_0 does not know (R_b, R_c, R_d) itself but $(\alpha(R_b), \alpha(R_c), \alpha(R_d))$ only. Thus, L_0 needs $(g, h, R_b, R_c, R_d, R_{\mathsf{A}_0})$ although K_0 can use $(g, h, \alpha(R_b), \alpha(R_c), \alpha(R_d), R_{\mathsf{A}_0})$ only.

In order to resolve this discrepancy, K_0 selects ρ_b, ρ_c, and ρ_d randomly, sets $R_b' = \beta(b, \rho_b)$, $R_c' = \beta(c, \rho_c)$, and $R_d' = \beta(d, \rho_d)$, and executes L_0 by feeding $(g, h, R_b', R_c', R_d', R_{\mathsf{A}_0})$. We will show that (R_b', R_c', R_d') is indistinguishable from (R_b, R_c, R_d) by using the random-like property of $\mathcal{G}$. Hence, we will be able to show that L_0 can output the correct discrete logarithm r_i even if L_0 is not fed (R_b, R_c, R_d) but (R_b', R_c', R_d'). Therefore, we will be able to show that K_0 is successful.

We now give the precise description of K_0. We describe how K_0 extract a plaintext from C_i, where i is an arbitrary number and C_i is the i-th encryption query of A_0. On input public key pk, ciphertext C_i, random tape R_{A_0} of A_0, the state st of K_0, and the random tape ρ_{K_0} of K_0, K_0 perform as follows. K_0 parses pk as (g, h, b, c, d), C_i as (u_i, v_i, e_i, π_i), and ρ_{K_0} as $\rho_b \| \rho_c \| \rho_d \| \rho_{\mathsf{L}_0}$, computes $R'_b = \beta(b; \rho_b)$, $R'_c = \beta(c; \rho_c)$, and $R'_d = \beta(d; \rho_d)$, sets $R'_{\mathsf{B}_0} = R'_b \| R'_c \| R'_d \| R_{\mathsf{A}_0}$, executes $\mathsf{L}_0(g, h, (u_i, v_i), R'_{\mathsf{B}_0}, \mathsf{st}; \rho_{\mathsf{L}_0})$, and obtains the output (r_i, st) of L_0. If $r_i = \mathsf{NonDH}$ holds, K_0 outputs $(\perp, \mathsf{st})$. Otherwise, K_0 computes $\theta_i = H(u_i, v_i, e_i)$ and $\pi'_i = (cd^{\theta_i})^{r_i}$. If $\pi_i = \pi'_i$ holds, K_0 outputs $(e_i / b^{r_i}, \mathsf{st})$. Otherwise, K_0 outputs $(\perp, \mathsf{st})$.

In order to show that K_0 is successful, we show that subroutine L_0 of K_0 can output the correct discrete logarithm with overwhelming probability even in the experiment $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}_0}_{\Pi,\mathsf{A}_0}(\lambda)$. To this end, we use the assumption that $\mathcal{G}$ is random-like. We construct an adversary C_0 for the computationally random-like property of $\mathcal{G}$. Let (a_*, R_*) be an instance of the game of computationally random-like property. C_0 would like to know whether $a_* = \alpha(R_*)$ holds or $R_* = \beta(a_*; \rho_*)$ holds for some ρ_*. $\mathsf{C}_0(a_*, R_*)$ selects $j_0 \in \{1, 2, 3\}$ randomly, selects R_j randomly and sets $a_j = \alpha(R_j)$ for $j < j_0$, sets $R_{j_0} = R_*$ and $a_{j_0} = a_*$, selects $a_j \in \mathcal{G}$ and a random tape ρ_j randomly, and sets $R_j = \beta(a_j; \rho_j)$ for $j > j_0$. Then C_0 sets $(b^\dagger, c^\dagger, d^\dagger, R^\dagger_b, R^\dagger_c, R^\dagger_d) = (a_1, a_2, a_3, R_1, R_2, R_3)$, randomly selects $g \in \mathcal{G}$, $x \in \mathbb{Z}_q$, and a random tapes R_{A_0} and ρ_{L_0}, sets $h = g^x$, $\mathsf{pk} = (g, h, b^\dagger, c^\dagger, d^\dagger)$, $\mathsf{st} = \varepsilon$, and $R^\dagger_{\mathsf{B}_0} = R_{\mathsf{A}_0} \| R^\dagger_b \| R^\dagger_c \| R^\dagger_d$, and executes $\mathsf{A}_0(\mathsf{pk}; R_{\mathsf{A}_0})$. If A_0 makes the i-th decryption query (u_i, v_i, e_i, π_i), C_0 executes $\mathsf{L}_0(g, h, (u_i, v_i), R^\dagger_{\mathsf{B}_0}, \mathsf{st}; \rho_{\mathsf{L}_0})$, and obtains the output (r_i, st) of L_0. If $r_i = \mathsf{NonDH}$ and $v_i = {u_i}^x$ hold, C_0 outputs 0 and terminates. If $r_i \in \mathbb{Z}_q$ and $(u_i, v_i) \neq (g^{r_i}, h^{r_i})$ hold, C_0 outputs 0 and terminates. Otherwise, C_0 computes $\theta_i = H(u_i, v_i, e_i)$ and $\pi'_i = (cd^{\theta_i})^{r_i}$. If $\pi_i = \pi'_i$ holds, C_0 sends e_i / b^{r_i} back to A_0. Otherwise, C_0 sends $\perp$ back to A_0. If A_0 terminates, C_0 outputs 1 and terminates.

If $R_* = \beta(a_*; \rho_*)$ and $j_0 = 1$ holds, the distribution of output of C_0 is the same as that of $\mathsf{DHK}^{\mathsf{L}_0}_{\mathcal{G},\mathsf{B}_0}(\lambda)$. If $a_* = \alpha(R_*)$ and $j_0 = 3$, the distribution of output of C_0 is the same as that of $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}_0}_{\Pi,\mathsf{A}_0}(\lambda)$. This means that L_0 outputs the correct discrete logarithm with overwhelming probability even in $\mathsf{EPA}^{\mathsf{Dec},\mathsf{K}_0}_{\Pi,\mathsf{A}_0}(\lambda)$.

We now show that $\mathsf{K}_0(\mathsf{pk}, C_i, R_{\mathsf{A}_0}, \mathsf{st}; \rho_{\mathsf{K}_0})$ outputs the correct answer with overwhelming probability. As before, we write pk as (g, h, b, c, d) and C_i as (u_i, v_i, e_i, π_i). Let (r_i, st) be the output of $\mathsf{L}_0(g, h, (u_i, v_i), R_{\mathsf{B}_0}, \mathsf{st}; \rho_{\mathsf{L}_0})$, $\mathsf{sk} = (z, z', x, x', y, y', \mathsf{pk})$ be the unknown secret key corresponding to pk and θ_i be $H(u_i, v_i, e_i)$.

We first consider the case where $r_i \neq \mathsf{NonDH}$. Since L_0 outputs the correct discrete logarithm with overwhelming probability, $(u_i, v_i) = (g^{r_i}, h^{r_i})$ holds with overwhelming probability. From the definition of the Cramer-Shoup encryption scheme, $\mathsf{Dec}_{\mathsf{sk}}(C_i)$ is equal to $e_i / {u_i}^z {v_i}^{z'}$ or $\perp$, depending on whether ${u_i}^{x+\theta_i y} {v_i}^{x'+\theta_i y'} = \pi_i$ holds or not. From $(u_i, v_i) = (g^{r_i}, h^{r_i})$, it follows that $e_i / {u_i}^z {v_i}^{z'} = e_i / g^{r_i z} h^{r_i z'} = e_i / b^{r_i}$ and ${u_i}^{x+\theta_i y} {v_i}^{x'+\theta_i y'} = g^{r_i x + r_i \theta_i y} h^{r_i x' + r_i \theta_i y'} = (g^x h^{x'})^r (g^y h^{y'})^{\theta_i r_i} = (cd^{\theta_i})^{r_i}$. Recall that K_0 outputs $e_i / {u_i}^z {v_i}^{z'}$ or $\perp$, depending on whether $(cd^{\theta_i})^{r_i} = e_i$ holds or not. This means that the output of K_0 is equal to $\mathsf{Dec}_{\mathsf{sk}}(C_0)$ with overwhelming probability.

We next consider the case where $r_i = \mathsf{NonDH}$. Since L_0 outputs the correct output with overwhelming probability, there is no $s \in \mathbb{Z}_q$ satisfying $(u_i, v_i) = (g^s, h^s)$. This means that there is no $(M, s) \in \mathcal{G} \times \mathbb{Z}_q$ satisfying $C_i = \mathsf{Enc}_{\mathsf{pk}}(M; s)$. Since the Cramer-Shoup scheme is regular, $\mathsf{Dec}_{\mathsf{sk}}(C_i) = \perp$ holds with overwhelming probability. Since the output of K_0 is $\perp$, the output of K_0 is equal to $\mathsf{Dec}_{\mathsf{sk}}(C_0)$ with overwhelming probability. □

4.3 The Reason We Succeed in Showing the Statistical PA-ness

The main theorem, Theorem 3, shows the *statistical* PA2-ness of the Cramer-Shoup scheme based on *computational* assumptions, such as the computationally random-like assumption and the DDH assumption. This seems strange at first glance. Hence, we here see why the statistical PA2-ness can be derived from computational assumptions. We proved the statistical PA2-ness as follows:

1. Prove that "EPA1 + computationally random-like ⇒ statistically PA2" (Theorem 11).
2. Prove that EPA1-ness of the Cramer-Shoup scheme from the DHK assumption.
3. Prove the computational random-like property of the Cramer-Shoup scheme from the computational assumptions.

The key point for proving the statistical PA2-ness is our new notion, the EPA1 security. In fact, we fail to prove the statistical PA2-ness of it, if we replace the EPA1-ness of Theorem 11 with the statistical PA1-ness. Recall that $X \simeq_{\mathrm{stat}} Y$ and $Y \simeq_{\mathrm{comp}} Z$ only implies $X \simeq_{\mathrm{comp}} Z$, where X, Y, and Z are random variables. Therefore, *statistical* PA1-ness + *computationally* random-like property only implies (at most) *computational* PA2 security.

In contrast, *EPA1* security + computationally random-like property implies the *statistical* PA2 security. The reason is as follow. Recall that the EPA1 (*Equality*-PA1) security means that "the *equality* $M = \mathsf{Dec}_{\mathsf{sk}}(C)$ holds with overwhelming probability," where M is an output of an extractor $\mathsf{K}(\cdots, C, \mathsf{List}, \cdots)$. Clearly, the computational indistinguishability changes the probability only negligibly. That is, if List' is *computationally* indistinguishable from List, the *equality* $M' = \mathsf{Dec}_{\mathsf{sk}}(C)$ also holds with overwhelming probability, where M' is an output of an extractor $\mathsf{K}(\cdots, C, \mathsf{List}', \cdots)$. (Here List is the list of random elements of $\mathcal{X}$ and List' is the list of encryptions. Their computational indistinguishability is ensured by the computational random-like property.) Hence, the *Equality*-PA1-ness + the *computationally* random-like implies the *Equality*-PA2-ness (and therefore implies *statistical* PA2-ness). Therefore, we can say that the EPA1 security allows us to show the statistical PA2-ness.

Another reason we can succeed in proving the statistical PA2-ness is in the definition of the DHK assumption. Recall that the DHK assumption ensures that an output of an extractor is not only indistinguishable but *equal* to the discrete logarithm with overwhelming probability. Hence, we can prove the *Equality*-PA1-ness of the Cramer-Shoup scheme under the DHK assumption and therefore can prove its statistical PA2-ness by using Theorem 11.

References

[BDPR98] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)

[BP04] Bellare, M., Palacio, A.: Towards plaintext-aware public-key encryption without random oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)

[BR94] Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)

[BD08] Birkett, J., Dent, A.W.: Relations Among Notions of Plaintext Awareness. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 47–64. Springer, Heidelberg (2008)

[B01] Boneh, D.: Simplified OAEP for the RSA and Rabin Functions. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 275–291. Springer, Heidelberg (2001)

[CS98] Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)

[CS01] Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes (2001)

[D91] Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)

[D06] Dent, A.W.: Cramer-Shoup is Plaintext-Aware in the Standard Model. In: EUROCRYPT 2006 (2006)

[F06] Fujisaki, E.: Plaintext Simulatability. IEICE Trans. Fundamentals, E89-A, pp.55-65. Preliminary version (2006), http://eprint.iacr.org/2004/218.pdf

[FO99] Fujisaki, E., Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost. In: Imai, H., Zheng, Y. (eds.) PKC 1999. LNCS, vol. 1560, pp. 53–68. Springer, Heidelberg (1999)

[FOPS01] Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP Is Secure under the RSA Assumption. In: CRYPTO 2001, pp. 260–274; J. Cryptology 17(2), pp.81-104 (2004)

[HLM03] Herzog, J., Liskov, M., Micali, S.: Plaintext Awareness via Key Registration. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 548–564. Springer, Heidelberg (2003)

[M01] Manger, J.: A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 230–238. Springer, Heidelberg (2001)

[S00] Shoup, V.: Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)

[S01] Shoup, V.: OAEP Reconsidered. In: CRYPTO 2001, pp.239–259 (2001); J. Cryptology, 15(4), 223–249 (2002)

[TO06] Teranishi, I., Ogata, W.: Relationship between Standard Model Plaintext Awareness and Message Hiding. In: ASIACRYPT 2006. IEICE Transactions 2008 91-A(1), pp.244-261

[TO08] Teranishi, I., Ogata, W.: Relationship between Two Approaches for Defining the Standard Model PA-ness. In: ACISP 2008 (2008)

General Certificateless Encryption and Timed-Release Encryption

Sherman S.M. Chow[1,⋆], Volker Roth[2], and Eleanor G. Rieffel[2]

[1] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[2] FX Palo Alto Laboratory
3400 Hillview Avenue
Palo Alto, CA 94304, USA
{vroth,rieffel}@fxpal.com

Abstract. While recent timed-release encryption (TRE) schemes are implicitly supported by a certificateless encryption (CLE) mechanism, the security models of CLE and TRE differ and there is no generic transformation from a CLE to a TRE. This paper gives a generalized model for CLE that fulfills the requirements of TRE. This model is secure against adversaries with adaptive trapdoor extraction capabilities, decryption capabilities for arbitrary public keys, and partial decryption capabilities. It also supports hierarchical identifiers. We propose a concrete scheme under our generalized model and prove it secure without random oracles, yielding the first strongly-secure security-mediated CLE and the first TRE in the standard model. In addition, our technique of partial decryption is different from the previous approach.

Keywords: Security-mediated certificateless encryption, timed-release encryption, standard model.

1 Introduction

In identity-based encryption (IBE) [29], encryption is done with respect to any arbitrary string viewed an identifier (ID). Since the birth of practical IBE constructions, this idea has been used to achieve other security goals, such as certificateless encryption (CLE) [1,14,16] and timed-release encryption (TRE) [3].

CLE is intermediate between IBE and traditional public key encryption (PKE). Traditional PKE requires a certification infrastructure but allows users to create their own public/private key pairs so that their private keys are truly private. Conversely, IBE avoids the need for certificates at the expense of adding a trusted key generation center (KGC) that generates the private keys, which means the KGC has the capability to decrypt all messages. CLE combines the

⋆ This research is done while the first author was a research intern of FXPAL. We thank Wolfgang Polak for helpful discussions and the reviewers for their feedback.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 126–143, 2008.

advantages of both: no certificates are needed and messages can only be decrypted by the recipient. Generally, CLE is constructed by combining IBE and PKE. The existence of the PKE component means that the KGC cannot decrypt messages. Instantaneous revocation is difficult for typical CLE schemes. Security-mediated certificateless encryption (SMCLE) addresses this issue.

In TRE, a message is encrypted under a public key and a time; both the private key and a time-dependent trapdoor are needed for decryption. A time-server is trusted to keep a trapdoor confidential until an appointed time. Apart from delayed release of information, TRE supports many other applications due to its small trapdoor size and its commitment provision (see [11,18]).

1.1 The Difficulty of Converting between CLE and TRE

A practical TRE requires system parameters to be small relative to the number of supported time periods. IBE supports an efficient time-based unlock mechanism by treating the identities as time periods [4,26]. This approach supports only universal disclosure of encrypted documents since one trapdoor can decrypt all ciphertexts for a specific time; the inherent key-escrow property of IBE prohibits the encryption for a designated receiver.

Since CLE is an "escrow-free version" of IBE, and both TRE and CLE are a kind of double-encryption, it is natural to think CLE is what we are looking for to realize a TRE. While most recent TRE schemes can be viewed as containing an implicit CLE mechanism, a generic transformation from CLE to TRE is unlikely to be provable secure [7]. Difficulty in reducing the confidentiality of TRE to that of CLE arises when the adversary is a "curious" time-server. In CLE, an identity is associated with only one public key, so a curious KGC is not allowed to replace the public key associated with an identifier arbitrarily (otherwise, decryption is trivial since it holds both parts of secrets). On the other hand, in TRE a time identifier is never bound to any public key, so the public key associated with a time identifier can be replaced. There is no way to simulate this implicit public key replacement when CLE is viewed as a black box.

There is another subtle difference in modeling of curious users. In a secure multi-user system, the security of a user is preserved even if other users are compromised. In CLE, the user secret key together with the trapdoor given by the KGC give the *full* private key. With the assumption that the user secret key will be securely deleted after the combination, most CLE models assume the adversary can get only trapdoors and full private keys. For most CLE schemes under this model (e.g. [17]), the user secret key cannot be recovered from the trapdoor and the full private key. Moreover, some CLE formulations [2,24,30] do not have user secret keys at all. In TRE, user secret keys are held by each user, which makes it impossible to reduce the security of TRE to that of CLE.

1.2 Our Contributions

Our generalized model for CLE overcomes the aforementioned difficulties and has sufficient power to fulfill the requirements of TRE. Our model is secure

against an adversary with adaptive trapdoor extraction capabilities for arbitrary identifiers (instead of selective identifiers, e.g. [4,27]), decryption capabilities for arbitrary public keys (as considered in strongly-secure CLE [17]) and partial decryption capabilities (as considered in security-mediated CLE [12]). Our model also supports hierarchical identifiers which have not been considered formally for CLE and TRE. Design choices behind our formulation are justified.

All previous concrete TRE schemes [3,7,8,9,10,15,18,21,23], and the only concrete SMCLE scheme [12], were proven in the random oracle model. Our model is strong but achievable: our proposed scheme is the first strongly-secure SMCLE. With our security-preserving transformation from a general CLE to a TRE, it also yields the first TRE in the standard model.

This work enriches the study of SMCLE by providing a novel partial decryption technique which is different from that in [12], and enriches TRE by supporting a new business model for the time-server. Finally, hierarchy of identifiers makes decryption of ciphertext for passed periods more manageable.

2 Related Work

2.1 Timed-Release Encryption

Early TRE schemes require interaction with the time-server. Rivest, Shamir and Wagner's idea [28] require senders to reveal the release-time of the messages in their interactions with the server, so the senders cannot be anonymous to the server. In Di Crescenzo, Ostrovsky and Rajaopalan 's scheme [15], it is the receiver who interacts with the time-server by invoking a "conditional oblivious transfer protocol", which is computationally intensive.

Blake and Chan made the first attempt to construct a non-interactive TRE [3]. The formal security model of message confidentiality was later considered independently by Cheon *et al.* [10] and Cathalo, Libert and Quisquater [7]. The former focuses on authenticated TRE. The latter also formalizes the release-time confidentiality. The recovery of past time-dependent trapdoors from a current trapdoor was studied in [9] and [26], which employs a hash chain and a tree structure [6] respectively. The study of the pre-open capability in TRE was initiated in [23] and improved by [18]. Recently, Chalkias, Hristu-Varsakelis and Stephanides proposed an efficient TRE scheme [8] with random oracles.

2.2 Certificateless Encryption

Al-Riyami and Paterson [1] proposed certificateless encryption in 2003. Extensive surveys of CLE security models and constructions can be found in [14,16]. Two types of adversaries are considered in certificateless encryption. A Type-I adversary models coalitions of rogue users without the master secret. Due to the lack of a certificate, the adversary is allowed to replace the public keys of users. A Type-II adversary models a curious KGC who has the master key but cannot replace the public keys of any users. In Al-Riyami and Paterson's security model

for encryption [1], a Type-I adversary can ask for the decryption of a ciphertext under a replaced public key. Schemes secure against such attacks are called "strongly-secure" [17], and the oracle is termed a "strong decryption oracle". A weaker type of adversary, termed Type-I$^-$, can only obtain a correct plaintext if the ciphertext is submitted along with the corresponding user secret key.

The Al-Riyami and Paterson scheme [1] is secure against both Type-I and Type-II adversaries in the random oracle model. It was believed [24,25,27] that [25] gave the first CLE in the standard model. However, it is possible to instantiate a prior generic construction in [12] with a PKE and an IBE in the standard model to obtain a secure CLE without random oracles. Both [25] and the instantiation of [12] are only secure against Type-I$^-$ attacks. Based on [19], a selective-ID secure CLE without random oracles was proposed [27]. This scheme cannot be efficiently extended to a TRE since the user's public key is dependent on the identity, which is never coupled with a fixed time-identifier in TRE. Recently, the first strongly-secure CLE in the standard model is proposed [17].

Al-Riyami and Paterson give an extension for hierarchical CLE [1]. However, no security model is given. We are not aware of any literature with formal work on hierarchical CLE, particularly none proven secure in the standard model.

Baek et al. proposed the first CLE that does not use pairings [2]. The CLE proposal [24] uses similar ideas, but their security proof ignores the public key replacement of the target user being attacked. This limitation is removed in Sun, Zhang and Baek's work [30]. To replace the pairing, these schemes make part of the user's public key dependent on the identity-specific trapdoor given by the KGC, which means TRE cannot be obtained trivially from these constructions.

Security-mediated certificateless encryption (SMCLE), introduced by Chow, Boyd and González Nieto [12], adds a security-mediator (SEM) who performs partial decryption for the user by request. This idea gives a more general treatment of the decryption queries in the CLE paradigm: the adversary can ask for partial decryption results under either the SEM trapdoor generated by the KGC or the user secret key A concrete construction in the random oracle model and a generic construction in the standard model are proposed in [12]. Prior to our work, no strongly-secure SMCLE existed in the standard model.

3 General Security-Mediated Certificateless Encryption

3.1 Notation

We use an ID-vector $\overrightarrow{ID} = (\mathsf{ID}_1, \mathsf{ID}_2, \cdots, \mathsf{ID}_L)$ to denote a hierarchy of identifiers $(\mathsf{ID}_1, \mathsf{ID}_2, \cdots, \mathsf{ID}_L)$. The length of $\overrightarrow{ID}$ is denoted by $|\overrightarrow{ID}| = L$. Let $\overrightarrow{ID}||\mathsf{ID}_r$ denote the vector $(\mathsf{ID}_1, \mathsf{ID}_2, \cdots, \mathsf{ID}_L, \mathsf{ID}_r)$ of length $|\overrightarrow{ID}| + 1$. We say that $\overrightarrow{ID}$ is a prefix of $\overrightarrow{ID'}$ if $|\overrightarrow{ID}| \leq |\overrightarrow{ID'}|$ and $\mathsf{ID}_i = \mathsf{ID}'_i$ for all $1 \leq i \leq |\overrightarrow{ID}|$. We use $\emptyset$ to denote an empty ID-vector where $|\emptyset| = 0$ and $\emptyset||\mathsf{ID}_r = \mathsf{ID}_r$. Finally, we use the notation $(\{0,1\}^n)^{\leq h}$ to denote the set of vectors of length less than or equal to h, where each component is a n-bit long bit-string.

3.2 Syntax

We propose a new definition of the (security-mediated) certificateless encryption, which also extends the definition of a 1-level SMCLE scheme in [12] to h levels.

Definition 1. *An h-level SMCLE scheme for identifiers of length n is defined by the following sextuple of* PPT *algorithms:*

- Setup *(run by the server) is a probabilistic algorithm which takes a security parameter 1^λ, outputs a master secret key Msk (which can also be denoted as $d_\emptyset$), and the global parameters Pub (which include $h = h(\lambda)$ and $n = n(\lambda)$ implicitly) We assume all other algorithms take Pub implicitly as an input.*
- $\mathsf{Extract}$ *(run by the server or any one who holds a trapdoor) is a possibly probabilistic algorithm which takes a trapdoor $d_{\overrightarrow{ID}}$ corresponding to an h-level identifier $\overrightarrow{ID} \in (\{0,1\}^n)^{\leq h}$, and a string $\mathsf{ID}_r \in \{0,1\}^n$, outputs a trapdoor key $d_{\overrightarrow{ID}||\mathsf{ID}_r}$ associated with the ID-vector $\overrightarrow{ID}||\mathsf{ID}_r$. The master secret key Msk is a trapdoor corresponding to a 0-level identifier.*
- KGen *(run by a user) is a probabilistic algorithm which generates a public/private key pair $(\mathsf{pk}_u, \mathsf{sk}_u)$.*
- Enc *(run by a sender) is a probabilistic algorithm which takes a message m from some implicit message space, an identifier $\overrightarrow{ID} \in (\{0,1\}^n)^{\leq h}$, and the receiver's public key pk_u as input , returns a ciphertext C.*
- Dec^S *(run by any one who holds the trapdoor, either a SEM in SMCLE or a receiver in CLE) is a possibly probabilistic algorithm which takes a ciphertext C and a trapdoor key $d_{\overrightarrow{ID}}$, returns either a token D which can be seen as a partial decryption, or an invalid flag $\perp$ (which is not in the message space).*
- Dec^U *(run by a receiver) is a possibly probabilistic algorithm which takes the ciphertext C, the receiver's secret key sk_u and a token D as input, returns either the plaintext, an invalid flag $\perp_D$ denoting D is an invalid token, or an invalid flag $\perp_C$ denoting the ciphertext is invalid.*

For correctness, we require that $\mathsf{Dec}^U(C, \mathsf{sk}, \mathsf{Dec}^S(C, \mathsf{Extract}(\mathsf{Msk}, \overrightarrow{ID}))) = m$ *for all* $\lambda \in \mathbb{N}$, *all* $(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, *all* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}$, *all message m, all ID-vector* $\overrightarrow{ID}$ *in* $(\{0,1\}^n)^{\leq h}$ *and all* $C \xleftarrow{\$} \mathsf{Enc}(m, \overrightarrow{ID}, \mathsf{pk})$.

3.3 Security

Each adversary has access to the following oracles:

1. An $\mathsf{ExtractO}$ oracle that takes an ID-vector $\overrightarrow{ID} \in (\{0,1\}^n)^{\leq h}$ as input and returns its trapdoor $d_{\overrightarrow{ID}}$.
2. An UskO oracle that takes a public key pk as input and returns its corresponding private key sk.
3. A DecO^S oracle that takes a ciphertext C and an ID-vector $\overrightarrow{ID}$, and outputs $\mathsf{Dec}^S(C, d_{\overrightarrow{ID}})$. Note that C may or may not be encrypted under $\overrightarrow{ID}$.
4. A DecO^U oracle that takes a ciphertext C, a public key pk and a token D, and outputs $\mathsf{Dec}^U(C, \mathsf{sk}, D)$ where sk is the secret key that matches pk.

5. A DecO oracle that takes a ciphertext C, an ID-vector $\overrightarrow{ID}$, and a public key pk; outputs $\mathsf{Dec}^U(C, \mathsf{sk}, D)$ where sk is the secret key that matches pk, $D = \mathsf{Dec}^S(C, d_{\overrightarrow{ID}})$ and C may or may not be encrypted under $\overrightarrow{ID}$ and pk.

Following common practice, we consider the two kinds of adversaries.

1. A Type-I adversary that models any coalition of rogue users, and who aims to break the confidentiality of another user's ciphertext.
2. A Type-II adversary that models a curious KGC, who aims to break the confidentiality of a user's ciphertext[1].

We use the common security model in which the adversary plays a two-phased game against a challenger. The game is modeled by the experiment below, $\mathsf{X} \in \{\mathrm{I}, \mathrm{II}\}$ denotes whether an PPT adversary $\mathcal{A} = (\mathcal{A}_{\mathsf{find}}, \mathcal{A}_{\mathsf{guess}})$ is of Type-I or II, and determines the allowed oracle queries $\mathcal{O}$ and the auxiliary data Aux.

Definition 2. Experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-X}}(\lambda)$

$(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathsf{pk}^*, \overrightarrow{ID}^*, \mathsf{state}) \xleftarrow{\$} \mathcal{A}_{\mathsf{find}}^{\mathcal{O}}(\mathsf{Pub}, \mathsf{Aux})$
$b \xleftarrow{\$} \{0,1\}$, $C^* \xleftarrow{\$} \mathsf{Enc}(m_b, \overrightarrow{ID}^*, \mathsf{pk}^*)$
$b' \xleftarrow{\$} \mathcal{A}_{\mathsf{guess}}^{\mathcal{O}}(C^*, \mathsf{state})$
If $(|m_0| \neq |m_1|) \vee (b \neq b')$ *then return 0 else return 1*

$\mathcal{O}$ *is a set of oracles* $\mathsf{ExtractO}(\cdot), \mathsf{UskO}(\cdot), \mathsf{DecO}^S(\cdot,\cdot), \mathsf{DecO}^U(\cdot,\cdot,\cdot), \mathsf{DecO}(\cdot,\cdot,\cdot)$.

Variables marked with * refer to challenges by the adversary. The adversary chooses a public key pk^* and an ID-vector $\overrightarrow{ID}^*$ to be challenged with, and the challenger returns a challenge ciphertext C^*. The following two definitions prohibit the adversary from trivially using the oracles to query for the answer to (parts of) the challenge.

Definition 3. *A hierarchical security-mediated certificateless encryption scheme is* (t, q_E, q_D, ϵ) *CCA-secure against a Type-I adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-I}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_E *extraction queries and* q_D *decryption queries (of any type), subjects to the following constraints:*

1. $\mathsf{Aux} = \emptyset$*, i.e. no auxiliary information is given to the adversary.*
2. *No* $\mathsf{ExtractO}(\overrightarrow{ID}')$ *query throughout the game, where* $\overrightarrow{ID}'$ *is a prefix of* $\overrightarrow{ID}^*$.
3. *No* $\mathsf{UskO}(\mathsf{pk})$ *query throughout the game for any* pk.
4. *No* $\mathsf{DecO}^S(C^*, \overrightarrow{ID}^*)$ *query throughout the game.*
5. *No* $\mathsf{DecO}(C^*, \overrightarrow{ID}^*, \mathsf{pk}^*)$ *query throughout the game.*

All public keys in the game are chosen by the adversary. It is natural to assume the adversary knows the corresponding secret keys.

[1] A rogue SEM is weaker than a Type-II adversary.

Definition 4. *A hierarchical security-mediated certificateless encryption scheme is* (t, q_K, q_D, ϵ) *CCA-secure against a Type-II adversary if* $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA-II}}(\lambda) = 1] - \frac{1}{2}| \leq \epsilon$ *for all* t*-time adversary* $\mathcal{A}$ *making at most* q_D *decryption queries (of any type), subjects to the following conditions:*

1. $\mathsf{Aux} = (\mathsf{Msk}, \{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\})$*, i.e.* $\mathcal{A}$ *is given the master secret and a set of challenge public keys.*
2. $\mathsf{pk}^* \in \{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\}$*, i.e. the challenge public key must be among the set given by the challenger.*
3. *No* $\mathsf{UskO}(\mathsf{pk})$ *query throughout the game if* $\mathsf{pk} \notin \{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\}$ *or* $\mathsf{pk} = \mathsf{pk}^*$*.*
4. *No* $\mathsf{DecO}^U(C^*, \mathsf{pk}^*, D)$ *query throughout the game, where* D *is outputted by the algorithm* $\mathsf{Dec}^S(C^*, d_{\overrightarrow{ID}^*})$*.*
5. *No* $\mathsf{DecO}(C^*, \overrightarrow{ID}^*, \mathsf{pk}^*)$ *query throughout the game.*

Since Msk is given to the adversary, the challenge public key must be in the set given by the challenger.

3.4 Discussions on Our Choices for Definition

This section explains the intuitions behind the choices made in formulating our definition and highlights the relationship between existing definitions and ours.

User key generation. In order to support more general applications like TRE, the interface for the algorithms needs a more general syntax. A subtle change is that our user key generation algorithm KGen only takes the system parameter as input but *not* the identifier. In some CLE schemes [2,24,27,30] the inclusion of the identifier, or the trapdoor for an identifier, is *essential* for the generation of the user public key. For these schemes, KGen can be executed only after $\mathsf{Extract}$, so straightforward adaption results in inefficient TREs in which the size of the user public key grows linearly with the number of supported time periods.

Simplification of Type-I adversary. In existing models for 1-level CLE [1,17], $\mathsf{ExtractO}$ query of $\overrightarrow{ID}^*$ is allowed; if such a query is issued, the challenge public key pk^* can no longer be chosen by the adversary. In our discussion, we separate this behavior from the Type-I model and consider this type of adversarial behavior ($\mathsf{ExtractO}(\overrightarrow{ID}')$ where $\overrightarrow{ID}'$ is a prefix of $\overrightarrow{ID}^*$) as a weaker variant of, and hence covered by, a Type-II adversary. It is true that our resulting definition for Type-I adversary is weaker, but the "missing part" is not omitted from the security requirement since CLEs must consider Type-II adversaries; this simplification was justified and adopted in [22, Section 2.3].

Existing models also allow full private key extraction for the public keys prepared by the challenger. In our Type-I game, all of the public keys to be attacked are generated by the adversary, so UskO query is prohibited. The remaining scenario, where the adversary intends to attack a public key given by the challenger, is also a weaker variant of our Type-II model. To conclude, we keep the essence of the existing models, and include the adversarially chosen public keys (for Type-I) and UskO (for Type-II) to match with TRE.

Strong decryption oracle. In our definition, the decryption oracle works even if the public key is adversarially chosen but the secret key is not supplied. The original definition of CLE [1] does not allow a strong decryption oracle for curious KGC adversary, but it is considered in recent work [17]. Adding the following restriction weakens Definition 4 to correspond to a Type-II$^-$ attack:

5. *(Type-II$^-$) No* $\mathsf{DecO}(C, \overrightarrow{ID}, \mathsf{pk})$ *query throughout the game for any* C *if* $\mathsf{pk} \notin \{\mathsf{pk}_1^*, \cdots \mathsf{pk}_{q_K}^*\}$*, unless the corresponding secret key* sk *is supplied when the* DecO *query is made.*

The Type-I$^-$ game can be obtained by adding $\mathsf{Aux} = \{\mathsf{pk}_1^*, \cdots \mathsf{pk}_{q_K}^*\}$ and the above restriction to Definition 3, but with a restriction on UskO as in Definition 4.

Implicit public key replacement. In our generalization of CLE, we "remove" (i.e. make implicit) the oracle for replacing the public key corresponding to an identifier. This change may affect the following choices:

1. The adversary's choice of the victim user it wishes to be challenged with,
2. The choice of user in decryption oracle queries.

However, there are other "interfaces" in our model such that the adversary can still make the above choices. Our model still allows the adversary to choose which identifier/public key it wants to attack. For decryption queries, the adversary can just supply different combination of identifier and public key to the DecO^S and DecO^U oracles. In this way, implicit replacement is done. In other words, when compared with the original model [1], the security model is not weakened, but generalized to cover applications of CLE such as TRE.

Reason for "removing" public key request and replacement oracles. In traditional definitions of CLE [1], oracles for retrieving and replacing public key depend upon the fact that an identifier is always bound to a particular user. Replacing a user's public key means changing the public key associated with a certain identifier. In TRE, identifiers correspond to policies governing the decryption, so a single identifier may be "shared" among multiple users. For this reason, our model must be free from the concept of "user = identifier".

Alternative definition of public key replacement. What about allowing a restricted public key replacement, such that a public key associated with an identifier can be replaced by a public key associated with another identifier, but not an arbitrary one supplied by the adversary? This definition still requires an identifier to belong to a single user. Moreover, this definition makes the treatment of a strong decryption oracle complicated: the idea of restricted replacement among a fixed set of public keys does not match well with decrypting under adversarially chosen public keys.

SMCLE is more general than plain CLE. The two separate decryption oracles in the SMCLE model provide a more general notion than CLE:

1. Some CLE schemes are not CCA-secure when the adversary has access to a partial decryption oracle (see [12]).

2. Since the decryption oracle is separated in two, the SMCLE model does not have the notion of a "full" private key which is present in previous CLE models (a full private key is a single secret for the complete decryption of the ciphertext). On the ground that separated secrets can always be concatenated into a single full one, this simplification (of private key) has already been adopted in more recent models [22].

Difference with the previous SMCLE definition. Our user decryption oracle DecO^U returns different invalid flags for the cases of invalid token from the SEM or invalid ciphertext. This distinction was not captured in [12].

User decryption oracle in SMCLE. To exclude trivial attacks, our Type-II adversary model disallows the challenge ciphertext C^* to be decrypted by the decryption oracle under the challenge public key and a token D obtained from the algorithm (not the oracle) $\mathsf{Dec}^S(C^*, \mathsf{ID}^*)$, where ID^* is the challenge identifier. To implement this restriction, our new SMCLE definition checks whether a token D is a *valid* token, corresponding to a ciphertext and an identifier.

While our security definition is tightly coupled with the ability to check the token, we think that it is natural for the user to be able to perform such a test (especially if the user pays for each token). Even without an explicit testing algorithm, the challenger may do the test as well since it simulates the scheme's execution. It gives a weaker definition if we prohibit any decryption query for the challenge ciphertext under the challenge public key, irrespective of the token.

Justifications for our definition of hierarchical CLE. In the hierarchical scheme of [1], an entity at level k derives a trapdoor for its children at level $k+1$ using both its trapdoor and its secret key. In our proposed model, a level k entity uses only the trapdoor obtained from its parent at level $k-1$ to derive keys for its children. We do not see any practical reason for requiring the secret key in the trapdoor derivation. Our definition avoids certain complications: for example, in [1], the decryption requires the public keys of all the ancestors.

We do allow the decryption of the ciphertext under $\overrightarrow{ID'}$ which is a prefix of $\overrightarrow{ID}^*$. This is stronger than the counterpart in some hierarchical IBE models [20].

Theorem 1. *If there exists a secure* 1*-level SMCLE scheme under Definition 3 and 4, there exists a CLE scheme which is secure under the definition of [1].*

Proof. Our aim is to build a simulator $\mathcal{B}$ which uses an adversary $\mathcal{A}$ of CLE to break the security of our 1-level SMCLE scheme. The simulator basically forwards everything (the system parameters, the oracle queries and responses, and the guess) back and forth between its own SMCLE challenger and the CLE adversary. Faced with a Type-II adversary of CLE, the simulator acts as a Type-II security of 1-level SMCLE. For a Type-I adversary of CLE, $\mathcal{B}$ flips a fair coin to determine its guess whether $\mathcal{A}$ will issue an $\mathsf{ExtractO}$ query of $\overrightarrow{ID}^*$. If it guesses not, $\mathcal{B}$ just plays the Type-I game as usual. If it guesses so, $\mathcal{B}$ will try to use $\mathcal{A}$ to win the Type-II game of SMCLE instead. The $\mathsf{ExtractO}$ query can be answered by $\mathcal{B}$ because it owns Msk now. The reduction tightness is reduced by a factor of 2. This simple trick is also used in [17, Appendix B, Game 4].

We omit the details for most queries, but focus on the distinctions that involve public key requests and replacement. The simulator must maintain a table to store the binding between an identifier and a public key. Whenever a Type-I adversary issues a public key request query, $\mathcal{B}$ executes $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}$, stores sk (so $\mathcal{B}$ can reply if $\mathcal{A}$ asks for it), and returns pk. For a Type-II adversary, $\mathcal{B}$ picks a random public key from $\{\mathsf{pk}_1^*, \cdots, \mathsf{pk}_{q_K}^*\}$ and assigns it as the public key of the queried ID. When $\mathcal{A}$ makes a key replacement query, the simulator updates its own table. For every other request regarding a particular identifier, the simulator retrieves the corresponding public key from its table and queries its own challenger accordingly. Finally, decryption queries of the CLE adversary are answered by combining results from the two partial decryption oracles. □

4 Our Proposed Construction

4.1 Preliminaries

Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative groups of prime order p for which there exists an efficiently computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that

1. *Bilinearity*: For all $u, v \in \mathbb{G}$ and $r, s \in \mathbb{Z}_p$, $\hat{e}(u^r, v^s) = \hat{e}(u, v)^{rs}$.
2. *Non-degeneracy*: $\hat{e}(u, v) \neq 1_{\mathbb{G}_T}$ for all $u, v \in \mathbb{G} \setminus \{1_{\mathbb{G}}\}$.

Our scheme's security relies on the intractability of the following problems:

Definition 5. *The* Decision **3**-Party Diffie-Hellman Problem *(3-DDH) in* $\mathbb{G}$ *is to decide if* $T = g^{\beta\gamma\delta}$ *given* $(g, g^\beta, g^\gamma, g^\delta, T) \in \mathbb{G}^5$. *Formally, defining the advantage of a* PPT *algorithm* $\mathcal{D}$, $Adv_{\mathcal{D}}^{3-\mathrm{DDH}}(\lambda)$, *as*

$$\begin{aligned} &| \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\beta, g^\gamma, g^\delta, T) | T \leftarrow g^{\beta\gamma\delta} \wedge \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p^*] \\ &- \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\beta, g^\gamma, g^\delta, T) | T \xleftarrow{\$} \mathbb{G} \wedge \beta, \gamma, \delta \xleftarrow{\$} \mathbb{Z}_p^*]|. \end{aligned}$$

We say 3-DDH is intractable if $Adv_{\mathcal{D}}^{3-\mathrm{DDH}}(\lambda)$ *is negligible in* λ *for all* PPT $\mathcal{D}$.

Compared with the Bilinear Diffie-Hellman (BDH) problem, the problem instance of 3-DDH is purely in $\mathbb{G}$ while that of BDH contains an element $\hat{t} \in \mathbb{G}_T$. If BDH problem is solvable, one can solve 3-DDH by feeding $(g, g^\beta, g^\gamma, g^\delta, \hat{e}(g, T))$ to a BDH oracle. The above assumption has been employed in [17].

We introduce a variant of the weak Bilinear Diffie-Hellman Inversion* (wBDHI*) assumption [4] below in the favor of 3-DDH. The original h-wBDHI* problem in $(\mathbb{G}, \mathbb{G}_T)$ [4] is to decide whether $\hat{t} = \hat{e}(g, g^\gamma)^{\alpha^{h+1}}$. The term "inversion" comes from the equivalence to the problem of deciding whether $\hat{t} = \hat{e}(g, g^\gamma)^{1/\alpha}$.

Definition 6. *The* h-Weak Diffie-Hellman Exponent Problem *(h-wDHE) in* $\mathbb{G}$ *is to decide if* $T = g^{\gamma\alpha^{h+1}}$ *given* $(g, g^\gamma, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^h}, T) \in \mathbb{G}^{h+3}$. *Formally, defining the advantage of a* PPT *algorithm* $\mathcal{D}$ *as*

$$\begin{aligned} Adv_{\mathcal{D}}^{h-\mathrm{wDHE}}(\lambda) = | \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\gamma, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^h}, T) | T \leftarrow g^{\gamma\alpha^{h+1}} \wedge \alpha, \gamma \xleftarrow{\$} \mathbb{Z}_p^*] \\ - \Pr[1 \xleftarrow{\$} \mathcal{D}(g, g^\gamma, g^\alpha, g^{\alpha^2}, \cdots, g^{\alpha^h}, T) | T \xleftarrow{\$} \mathbb{G} \wedge \alpha, \gamma \xleftarrow{\$} \mathbb{Z}_p^*]|. \end{aligned}$$

We say h-wDHE is intractable if $Adv_{\mathcal{D}}^{h-\mathrm{wDHE}}(\lambda)$ *is negligible in* λ *for all* PPT $\mathcal{D}$.

We require a family of collision resistant hash functions $\mathcal{H}$ too.

Definition 7. *A hash function* $H \xleftarrow{\$} \mathcal{H}(\lambda)$ *is* collision resistant *if*

$$Adv_{\mathcal{C}}^{\mathrm{CR}}(\lambda) = \Pr[H(x) = H(y) \wedge x \neq y | (x, y) \xleftarrow{\$} \mathcal{C}(1^{\lambda}, H) \wedge H \xleftarrow{\$} \mathcal{H}(\lambda)]$$

is negligible as a function of the security parameter λ *for all* PPT *algorithms* $\mathcal{C}$.

4.2 Proposed Construction

Our construction is an h-level generalization of the concrete construction for 1-level in [17]. While [17] uses the technique of [5] to achieve strong decryption oracle, we use the same technique for a different purpose, which is a new way (other than the only known way in [12]) to support partial decryption oracle.

$\mathsf{Setup}(1^{\lambda}, n)$: Let $\mathbb{G}$, $\mathbb{G}_T$ be two multiplicative groups with a bilinear map $\hat{e}$ as defined before. They are of the same order p, which is a prime and $2^{\lambda} < p < 2^{\lambda+1}$.

- **Encryption key:** choose two generators $g, g_2 \in_R \mathbb{G}$.
- **Master public key:** choose an exponent $\alpha \in_R \mathbb{Z}_p$ and set $g_1 = g^{\alpha}$.
- **Hash key for identifier-based key derivation:** choose h many $(\ell+1)$-length vectors $\vec{U}_1, \cdots, \vec{U}_h \in_R \mathbb{G}^{\ell+1}$, where each $\vec{U}_j = (u'_j, u_{j,1}, \cdots, u_{j,\ell})$, $1 \leq j \leq h$. ℓ is a tunable parameter which is a factor of n and $1 \leq \ell \leq n$. Each vector $\vec{U}_j$ $(1 \leq j \leq h)$ corresponds to the j-th level of the hierarchy. We use the notation $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_j, \cdots, \mathsf{ID}_k)$ to denote a hierarchy of k n-bit string ID_j's. We write ID_j as ℓ blocks each of length n/ℓ bits $(\mathsf{ID}_{j,1}, \cdots, \mathsf{ID}_{j,\ell})$. We define $F_{\vec{U}_j}(\mathsf{ID}_j) = u'_j \prod_{i=1}^{\ell} u_{j,i}^{\mathsf{ID}_{j,i}}$.
- **Hash key for ciphertext validity:** choose an $(n+1)$-length vector $\vec{V} = (v', v_1, \cdots, v_n) \in_R \mathbb{G}^{n+1}$. This vector defines the hash function $F_{\vec{V}}(w) = v' \prod_{j=1}^{n} v_j{}^{b_j}$ where w is a n-bit string $b_1 b_2 \cdots b_n$.
- **Hash function:** pick a function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ from a family of collision-resistant hash functions according to the parameter λ.

The public parameters Pub and the master secret key Msk are given by

$$\mathsf{Pub} = (\lambda, p, \mathbb{G}, \mathbb{G}_T, \hat{e}(\cdot,\cdot), n, \ell, g, g_1, g_2, \vec{U}_1, \cdots, \vec{U}_h, \vec{V}, H(\cdot)), \qquad \mathsf{Msk} = g_2^{\alpha}.$$

We require the discrete logarithms (with respect to g) of all $\mathbb{G}$ elements in Pub except g, g_1 to be unknown to the KGC. In practice, these elements can be generated from a pseudorandom function of a public seed.

$\mathsf{Extract}(d_{\overrightarrow{ID}}, \mathsf{ID}_r)$: For $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_k)$ for $k \leq h$, a trapdoor is in the form:

$$d_{\overrightarrow{ID}} = (a_1, a_2, \vec{z}_{k+1}, \cdots, \vec{z}_h) = (g_2^{\alpha} \cdot (\prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j))^r, g^r, (\vec{U}_{k+1})^r, \cdots, (\vec{U}_h)^r),$$

where $r \in_R \mathbb{Z}_p^*$ and $(\vec{U}_j)^r = ((u'_j)^r, (u_{j,1})^r, \cdots, (u_{j,\ell})^r)$.

Note that (a_1, a_2) is sufficient for decryption, while $\vec{z}_{k+1}, \cdots, \vec{z}_h$ can help the derivation of the trapdoor for $(\mathsf{ID}_1, \cdots, \mathsf{ID}_k, \mathsf{ID}_{k+1})$ for any n-bit string ID_{k+1} and $k+1 \leq h$. To generate $d_{\overrightarrow{ID}||\mathsf{ID}_r}$ parse $d_{\overrightarrow{ID}} = (a_1, a_2, (z_{k+1}, z_{k+1,1}, \cdots, z_{k+1,\ell}), \cdots, (z_h, z_{h,1}, \cdots, z_{h,\ell}))$ and parse ID_r as ℓ blocks $(\mathsf{ID}_{r,1}, \cdots, \mathsf{ID}_{r,\ell})$ where each block is of length n/ℓ bits, pick $t \in_R \mathbb{Z}_p^*$ and output where the multiplication of

$$d_{\overrightarrow{ID}||\mathsf{ID}_r} = (a_1 \cdot z_{k+1} \prod_{i=1}^{\ell} (z_{k+1,i})^{\mathsf{ID}_{r,i}} \cdot (\prod_{j=1}^{k+1} F_{\vec{U}_j}(\mathsf{ID}_j))^t, a_2 \cdot g^t, \vec{z}_{k+2} \cdot (\vec{U}_{k+2})^t \cdots, \vec{z}_h \cdot (\vec{U}_h)^t$$

two vectors are defined component-wise, i.e. $\vec{z}_j \cdot \vec{\nu}_j = (z_j \cdot \nu_j, z_{j,1} \cdot \nu_{j,1}, \cdots, z_{j,\ell} \cdot \nu_{j,\ell})$. $d_{\overrightarrow{ID}}$ becomes shorter as the length of $\overrightarrow{ID}$ increases.

$\mathsf{KGen}()$: Pick $\mathsf{sk} \in_R \mathbb{Z}_p^*$, return $\mathsf{pk} = (X, Y) = (g^{\mathsf{sk}}, g_1^{\mathsf{sk}})$ and sk as the key pair.

$\mathsf{Enc}(m, \overrightarrow{ID}, \mathsf{pk})$: To encrypt $m \in \mathbb{G}_T$ for $\overrightarrow{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_k)$ where $k \leq h$, parse pk as (X, Y), then check that it is a valid public key by verifying[2] that $\hat{e}(X, g_1) = \hat{e}(g, Y)$. If equality holds, pick $s \in_R \mathbb{Z}_p^*$ and compute

$$C = (C_1, C_2, \tau, \sigma) = (m \cdot \hat{e}(Y, g_2)^s, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j)^s, g^s, F_{\vec{V}}(w)^s)$$

where $w = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$.

$\mathsf{Dec}^S(C, d_{\overrightarrow{ID}})$: Parse C as (C_1, C_2, τ, σ), and $d_{\overrightarrow{ID}}$ as $(a_1, a_2, \cdots)$. First check if $\hat{e}(\tau, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j) \cdot F_{\vec{V}}(w')) = \hat{e}(g, C_2 \cdot \sigma)$ where $w' = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$. Return $\perp$ if inequality holds or any parsing is not possible, otherwise pick $t \in_R \mathbb{Z}_p^*$ and return

$$D = (D_1, D_2, D_3) = (a_1 \cdot F_{\vec{V}}(w')^t, a_2, g^t).$$

$\mathsf{Dec}^U(C, \mathsf{sk}, D)$: Parse C as (C_1, C_2, τ, σ) and check if $\hat{e}(\tau, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j) \cdot F_{\vec{V}}(w')) = \hat{e}(g, C_2 \cdot \sigma)$ where $w' = H(C_1, C_2, \tau, \overrightarrow{ID}, \mathsf{pk})$. If equality does not hold or parsing is not possible, return $\perp_C$. Next, parse D as (D_1, D_2, D_3) and check if $\hat{e}(g, D_1) = \hat{e}(g_1, g_2)\hat{e}(D_2, \prod_{j=1}^{k} F_{\vec{U}_j}(\mathsf{ID}_j))\hat{e}(D_3, F_{\vec{V}}(w'))$[3]. If equality does not hold or parsing is not possible, return $\perp_D$. Otherwise, return

$$m \leftarrow C_1 \cdot \left(\frac{\hat{e}(C_2, D_2)\hat{e}(\sigma, D_3)}{\hat{e}(\tau, D_1)} \right)^{\mathsf{sk}}.$$

[2] One pairing computation can be saved by a trick adopted in [17]: pick $\xi \in_R \mathbb{Z}_p^*$ and compute $C_1 = m \cdot \hat{e}(Y, g_2 \cdot g^{\xi})^s / \hat{e}(X, g_1^{s\xi})$.

[3] The same trick for minimizing the number of pairing computations involved in checking the ciphertext and the token can be incorporated to the final decryption step. The modified decryption algorithm only uses 4 pairing computations; however, it gives a random message (instead of an invalid flag $\perp$) for an invalid ciphertext.

4.3 Analysis

Similar to [4], the ciphertext size of our scheme is independent of the hierarchy length. This is also beneficial when it is used as a TRE (see Section 5.5).

In the concrete SMCLE scheme of Chow, Boyd and González Nieto [12], partial decryption uses the pairing function $\hat{e}(\cdot,\cdot)$ to pair part of the ciphertext and the ID-based private key. To make this partial decryption result verifiable requires turning a generic interactive proof-of-knowledge non-interactive. Our scheme employs a different technique such that the token generated by the partial decryption is publicly and non-interactively verifiable.

Our scheme's security is asserted by Theorem 2; [13] contains a proof.

Theorem 2. *Our scheme is secure against Type-I attack and Type-II attack (Definition 3 and 4) if h-wDHE problem and 3-DDH problem is intractable.*

5 Applying General Certificateless Encryption to TRE

5.1 Syntax of Timed-Release Encryption

For ease of discussion, consider only 1-level of time-identifiers as in [7]. It can be shown that our results hold for an h-level analog.

Definition 8. *A TRE scheme for time-identifiers of length n (n is a polynomially-bounded function) is defined by the following sextuple of* PPT *algorithms:*

- Setup *(run by the server) is a probabilistic algorithm which takes a security parameter 1^λ, outputs a master secret key* Msk*, and the global parameters* Pub*. We assume that λ and $n = n(\lambda)$ are implicit in* Pub *and all other algorithms take* Pub *implicitly as an input.*
- $\mathsf{Extract}$ *(run by the server) is a possibly probabilistic algorithm which takes the master secret key* Msk *and a string* $\mathsf{ID} \in \{0,1\}^n$*, outputs a trapdoor key d_{ID} associated with the identifier* ID*.*
- KGen *(run by a user) is a probabilistic algorithm which generates a public/private key pair* $(\mathsf{pk}_u, \mathsf{sk}_u)$*.*
- Enc *(run by a sender) is a probabilistic algorithm which takes a message m from some implicit message space, an identifier* $\mathsf{ID} \in \{0,1\}^n$*, and the receiver's public key* pk_u *as input, returns a ciphertext C.*
- Dec^S *(run by any one who holds the trapdoor, either a SEM or a receiver) is a possibly probabilistic algorithm which takes a ciphertext C and a trapdoor key d_{ID} as input, returns either a token D which can be seen as a partial decryption of C, or an invalid flag $\perp$ (which is not in the message space).*
- Dec^U *(run by a receiver) is a possibly probabilistic algorithm which takes the ciphertext C, the receiver's secret key* sk_u *and a token D as input, returns either the plaintext, an invalid flag $\perp_D$ denoting D is an invalid token, or an invalid flag $\perp_C$ denoting the ciphertext is invalid.*

For correctness, we require that $\mathsf{Dec}^U(C, \mathsf{sk}, \mathsf{Dec}^S(C, \mathsf{Extract}(\mathsf{Msk}, \mathsf{ID}))) = m$ *for all* $\lambda \in \mathbb{N}$*, all* $(\mathsf{Pub}, \mathsf{Msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}$*, all message* m*, all identifier* ID *in* $\{0,1\}^n$ *and all* $C \xleftarrow{\$} \mathsf{Enc}(m, \mathsf{ID}, \mathsf{pk})$.

5.2 Timed-Release Encryption from Certificateless Encryption

Given a SMCLE scheme $\{\mathcal{SMC}.\mathsf{Setup}, \mathcal{SMC}.\mathsf{Extract}, \mathcal{SMC}.\mathsf{KGen}, \mathcal{SMC}.\mathsf{Enc}, \mathcal{SMC}.\mathsf{Dec}^S, \mathcal{SMC}.\mathsf{Dec}^U\}$, a TRE scheme $\{\mathcal{TRE}.\mathsf{Setup}, \mathcal{TRE}.\mathsf{Extract}, \mathcal{TRE}.\mathsf{KGen}, \mathcal{TRE}.\mathsf{Enc}, \mathcal{TRE}.\mathsf{Dec}^S, \mathcal{TRE}.\mathsf{Dec}^U\}$ can be built as below.

$\mathcal{TRE}.\mathsf{Setup}(1^\lambda, n)$: Given a security parameter λ and the length of the time-identifier n, execute $(\mathsf{Msk}, \mathsf{Pub}) \leftarrow \mathcal{SMC}.\mathsf{Setup}(1^\lambda, n)$, retain Msk as the master secret key and publish Pub as the global parameters.

$\mathcal{TRE}.\mathsf{Extract}(\mathsf{Msk}, \mathsf{ID})$: For a time-identifier $\mathsf{ID} \in \{0,1\}^n$, the time-server returns $d_{\mathsf{ID}} \leftarrow \mathcal{SMC}.\mathsf{Extract}(\mathsf{Msk}, \mathsf{ID})$.

$\mathcal{TRE}.\mathsf{KGen}()$: Return $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{SMC}.\mathsf{KGen}()$ as the user's key pair.

$\mathcal{TRE}.\mathsf{Enc}(m, \mathsf{ID}, \mathsf{pk})$: To encrypt $m \in \mathbb{G}_T$ for pk under the time $\mathsf{ID} \in \{0,1\}^n$, return $\mathcal{SMC}.\mathsf{Enc}(m, \mathsf{ID}, \mathsf{pk})$, which may be $\perp$ if pk is an invalid public key.

$\mathcal{TRE}.\mathsf{Dec}^S(C, d_{\mathsf{ID}})$: To partially decrypt C by a time-dependent trapdoor d_{ID}, return $D \leftarrow \mathcal{SMC}.\mathsf{Dec}^S(C, d_{\mathsf{ID}})$.

$\mathcal{TRE}.\mathsf{Dec}^U(C, \mathsf{sk}, D)$: To decrypt C by the secret key sk and the token D, just return $\mathcal{SMC}.\mathsf{Dec}^U(C, \mathsf{sk}, D)$.

Theorem 3. *If* $\mathcal{SMC}$ *is an* 1*-level SMCLE scheme which is CCA-secure against Type-I adversary (Definition 3),* $\mathcal{TRE}$ *is CCA-secure against Type-I adversary.*

Theorem 4. *If* $\mathcal{SMC}$ *is an* 1*-level SMCLE scheme which is CCA-secure against Type-II adversary (Definition 4),* $\mathcal{TRE}$ *is CCA-secure against Type-II adversary.*

Proof. The security models of TRE can be found in [13]. We prove by contradiction. Suppose $\mathcal{A}$ is a Type-X adversary such that $|\Pr[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{CCA}'-\mathsf{X}}(\lambda) = 1] - \frac{1}{2}| > \epsilon$, we construct an adversary $\mathcal{B}$ with $|\Pr[\mathbf{Exp}_{\mathcal{B}}^{\mathsf{CCA}-\mathsf{X}}(\lambda) = 1] - \frac{1}{2}| > \epsilon$ in the face of a SMCLE challenger $\mathcal{C}$ where the running times of $\mathcal{B}$ and $\mathcal{A}$ are equal.

Setup: When $\mathcal{C}$ gives $\mathcal{B}$ $(\mathsf{Pub}, \mathsf{Aux})$, $\mathcal{B}$ just forwards it to $\mathcal{A}$. The public key to be passed to $\mathcal{A}$ is either chosen from the a set of public key in Aux (in Type-II game), or chosen by $\mathcal{B}$ itself (in Type-I game).

First Phase of Queries: $\mathcal{B}$ forwards every request of $\mathcal{A}$ to the oracles of its own challenger $\mathcal{C}$. From the description of $\mathcal{TRE}$, we can see that every legitimate oracle query made by $\mathcal{A}$ can be answered faithfully.

Challenge: When $\mathcal{A}$ gives $\mathcal{B}$ $(m_0, m_1, \mathsf{pk}^*, \mathsf{ID}^*)$, $\mathcal{B}$ just forwards it to $\mathcal{C}$.

Second Phase of Queries: Again, $\mathcal{B}$ just forwards every request of $\mathcal{A}$ to the oracles of its own challenger $\mathcal{C}$. From the description of $\mathcal{TRE}$, it is easy to see that every oracle query which does not violate the restriction enforced by $\mathcal{A}$ also does not violate the restriction enforced by $\mathcal{C}$.

Output: Finally, $\mathcal{A}$ outputs a bit b, $\mathcal{B}$ forwards it to $\mathcal{C}$ as its own answer. The probability for $\mathcal{A}$ to win the TRE experiment simulated by $\mathcal{B}$ is equal to the probability for $\mathcal{B}$ to win the SMCLE game played against $\mathcal{C}$. It is easy to see that the running times of $\mathcal{A}$ and $\mathcal{B}$ are the same. □

These theorems show that the scheme presented in section 4 can be instantiated as a TRE scheme without a random oracle.

5.3 Certificateless Encryption from Timed-Release Encryption

One may expect that a general CLE can be constructed from any TRE. The usage of time-identifiers, however, is only one specific instantiation of the timed-release idea. Other formulations of TRE, different from Definition 8, exist; for example, in [9], time is captured by the number of repeated computations of one-way hash function. Also, the notion of CLE supports an exponential number of arbitrary identifiers[4], so a CLE scheme cannot be realized by a TRE if the total number of time periods supported is too few.

There is an important difference in the definitions of security between CLE and TRE: the public keys in TRE are *certified* while there is no certification in CLE, so public keys can be chosen adversarially. Typically in TRE [3,8,10,18,23], a single public key is *given* to the adversary as the target of attack. However, the non-standard TRE formulation in [7] does allow uncertified public keys.

5.4 Security-Mediator in Timed-Release Encryption

The introduction of a security-mediator to the TRE paradigm gives a new business model for the time-server due to the support for partial decryption. Traditional TRE allows the time-server to release only a system-wide time-dependent trapdoor. The time-server can charge for each partial decryption request of a ciphertext by the time-dependent trapdoor; the partial decryption of one ciphertext would not help the decryption of any other ciphertext.

5.5 Time Hierarchy

Since each identifier corresponds to a single time period, the server must publish t private keys once t time-periods have passed. The amount of data that must be posted can be reduced given a hierarchical CLE by using the Canetti, Halevi

[4] Even though the scheme may be insecure when more than a polynomial number of trapdoors are compromised by a single adversary.

and Katz (CHK) forward secure encryption [6] in reverse [4]. For a total of T time periods, the CHK framework is set up as a tree of depth $\log T$. To encrypt a message for time $t < T$, the time identifier is the CHK identifier for time period $T-t$. Release of trapdoor is done in the same manner: the private key for the time period $T - t$ is released on the t^{th} time period. This single private key enables anyone to derive the private keys for CHK time periods $T - t, T - t + 1, \cdots, T$, so the user can obtain trapdoors for times $1, \cdots, t$. This trick enables the server to publish only a single private key of $O(\log^2 T)$ group elements at any time.

6 Conclusions

Cryptographers seek and try to achieve the strongest possible security definition. Previous models of certificateless encryption (CLE) were too restrictive: they could not give the desired security properties when instantiated as timed-release encryption (TRE). Our generalized CLE model supports the requirements of TRE; all future CLE proposals in our general model automatically give secure TRE schemes. Our model is defined against full-identifier extraction, decryption under arbitrary public key, and partial decryption, to achieve strong security. Our concrete scheme yields the first strongly-secure (hierarchical) security-mediated CLE and the first TRE in the standard model.

References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003), `http://eprint.iacr.org/2003/126`
2. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless Public Key Encryption Without Pairing. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005)
3. Blake, I.F., Chan, A.C.-F.: Scalable, Server-Passive, User-Anonymous Timed Release Cryptography. In: ICDCS 2005, pp. 504–513. IEEE Computer Society, Los Alamitos (2005)
4. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
5. Boyen, X., Mei, Q., Waters, B.: Direct Chosen Ciphertext Security from Identity-based Techniques. In: ACM CCS 2005, pp. 320–329 (2005)
6. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. Journal of Cryptology 20(3), 265–294 (2007)
7. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and Non-interactive Timed-Release Encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
8. Chalkias, K., Hristu-Varsakelis, D., Stephanides, G.: Improved Anonymous Timed-Release Encryption. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 311–326. Springer, Heidelberg (2007)
9. Chalkias, K., Stephanides, G.: Timed Release Cryptography from Bilinear Pairings Using Hash Chains. In: Leitold, H., Markatos, E.P. (eds.) CMS 2006. LNCS, vol. 4237, pp. 130–140. Springer, Heidelberg (2006)

10. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Timed-Release and Key-Insulated Public Key Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 191–205. Springer, Heidelberg (2006)
11. Chow, S.S.M.: Token-Controlled Public Key Encryption in the Standard Model. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 315–332. Springer, Heidelberg (2007)
12. Chow, S.S.M., Boyd, C., González-Nieto, J.M.: Security-Mediated Certificateless Cryptography. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 508–524. Springer, Heidelberg (2006)
13. Chow, S.S.M., Roth, V., Rieffel, E.G.: General Certificateless Encryption and Timed-Release Encryption. Cryptology ePrint Archive, Report 2008/023 (2008) (Full Version)
14. Chow, S.S.M.: Certificateless Encryption. In: Identity-Based Cryptography. IOS Press, Amsterdam (to appear, 2008)
15. Crescenzo, G.D., Ostrovsky, R., Rajagopalan, S.: Conditional Oblivious Transfer and Timed-Release Encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
16. Dent, A.W.: A Survey of Certificateless Encryption Schemes and Security Models. Cryptology ePrint Archive, Report 2006/211 (2006)
17. Dent, A.W., Libert, B., Paterson, K.G.: Certificateless Encryption Schemes Strongly Secure in the Standard Model. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 344–359. Springer, Heidelberg (2008), http://eprint.iacr.org/2007/121
18. Dent, A.W., Tang, Q.: Revisiting the Security Model for Timed-Release Public-Key Encryption with Pre-Open Capability. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 158–174. Springer, Heidelberg (2007)
19. Gentry, C.: Practical Identity-Based Encryption Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
20. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
21. Hristu-Varsakelis, D., Chalkias, K., Stephanides, G.: Low-cost Anonymous Timed-Release Encryption. In: Symposium on Information Assurance and Security, pp. 77–82. IEEE Computer Society, Los Alamitos (2007)
22. Hu, B.C., Wong, D.S., Zhang, Z., Deng, X.: Certificateless Signature: A New Security Model and An Improved Generic Construction. Designs, Codes and Cryptography 42(2), 109–126 (2007)
23. Hwang, Y.H., Yum, D.H., Lee, P.J.: Timed-Release Encryption with Pre-open Capability and Its Application to Certified E-mail System. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 344–358. Springer, Heidelberg (2005)
24. Lai, J., Kou, W.: Self-Generated-Certificate Public Key Encryption Without Pairing. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 476–489. Springer, Heidelberg (2007)
25. Liu, J.K., Au, M.H., Susilo, W.: Self-Generated-Certificate Public Key Cryptography and Certificateless Signature / Encryption Scheme in the Standard Model. In: ASIACCS 2007. ACM, New York (2007)
26. Nali, D., Adams, C.M., Miri, A.: Hierarchical Time-based Information Release. International Journal of Information Security 5(2), 92–104 (2006)

27. Park, J.H., Choi, K.Y., Hwang, J.Y., Lee, D.H.: Certificateless Public Key Encryption in the Selective-ID Security Model (Without Random Oracles). In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 60–82. Springer, Heidelberg (2007)
28. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock Puzzles and Timed-release Crypto. Technical Report MIT/LCS/TR-684, Massachusetts Institute of Technology (1996)
29. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
30. Sun, Y., Zhang, F., Baek, J.: Strongly Secure Certificateless Public Key Encryption Without Pairing. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 194–208. Springer, Heidelberg (2007)

Efficient Certificate-Based Encryption in the Standard Model*

Joseph K. Liu and Jianying Zhou

Cryptography and Security Department
Institute for Infocomm Research
Singapore
{ksliu,jyzhou}@i2r.a-star.edu.sg

Abstract. In this paper, we propose a new Certificate-Based Encryption (CBE) scheme which is fully secure in the standard model. We achieve chosen ciphertext (CCA) security directly without any transformation. When compared to all previous generic constructions (in either random oracle or standard model), our scheme is far more efficient than those schemes. When compared to the CBE scheme in [16] (which is the only concrete implementation secure in the standard model), we enjoy a great improvement in terms of space efficiency. Their scheme requires more than 160 group elements for the public parameters in order to gain an acceptable security. Our scheme just requires 5 group elements. In addition, the message space of our scheme is almost double as the one in [16]. A larger message space implies that it requires a smaller number of encryption operations of the same plaintext, resulting in a smaller overall ciphertext and overhead as well.

1 Introduction

Public Key Infrastructure (PKI). In traditional public key cryptography (PKC), a user Alice signs a message using her private key. A verifier Bob verifies the signature using Alice's public key. However, the public key is just a random string and it does not provide authentication of the signer by itself. This problem can be solved by using a certificate generated by a trusted party called the Certificate Authority (CA) that provides an unforgeable signature and trusted link between the public key and the identity of the signer. The hierarchical framework is called public key infrastructure (PKI) to issue and manage certificate (chain). In this case, before the verification of a signature, Bob needs to obtain Alice's certificate in advance and verify the validity of her certificate. If it is valid, Bob extracts the corresponding public key which is then used to verify the signature. In the point of view of a verifier, it takes two verification steps for independent signatures. It seems not efficient and not practical enough, especially when the number of users is very large.

* This work is partially funded by the EU project SMEPP-033563.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 144–155, 2008.

Identity-Based cryptography (IBC). Identity-based cryptography (IBC), invented by Shamair [17] in 1984, solves this problem by using Alice's identity (or email address) which is an arbitrary string as her public key while the corresponding private key is a result of some mathematical operation that takes as input the user's identity and the master secret key of a trusted authority, referred as "Private Key Generator (PKG)". In this way, the certificate is implicitly provided and it is no longer necessary to explicitly authenticate public keys. The main disadvantage of identity-based cryptography is an unconditional trust to the PKG. This is even worse than traditional PKC since the secret key of every user is generated by the PKG, it can impersonate any user, or decrypt any ciphertext.

Certificate-Based cryptography (CBC). To integrate the merits of IBC into PKI, Gentry [10] introduced the concept of Certificate-Based encryption (CBE). A CBE scheme combines a public key encryption scheme and an identity based encryption scheme between a certifier and a user. Each user generates his own private and public key and request a certificate from the CA while the CA uses the key generation algorithm of an identity based encryption (IBE) [5] scheme to generate certificate. Unlike traditional PKI, the certificate in CBC is implicitly used as part of the user private key for decryption, which requires both the user-generated private key and the certificate. Although the CA knows the certificate, it does not have the user private key. Thus it cannot decrypt anything. In addition to encryption, several certificate-based signature schemes [12,13,15] and ring signature scheme [4] were also proposed.

In parallel to CBC, certificateless cryptography [1] and self-generated-certificate public key cryptography [14] are another solutions to the key escrow problem inherited by IBC.

1.1 Related Works

The original scheme of Gentry relied on the original identity-based encryption (IBE) scheme of Boneh-Franklin [5] and then on the Fujisaki-Okamoto transform [8] to obtain full security in the random oracle model. Some generic constructions were proposed in [18,7] for constructing a CBE from an IBE (although Yum-Lee construction [18] was broken by Galindo *et al.* [9]) while another generic construction was given in [2] from a certificateless encryption (CLE) scheme. A concrete construction in the standard model was also proposed in [16].

1.2 Contribution

In this paper, we propose a new CBE scheme that is fully chosen ciphertext (CCA) secure in the standard model. Although there are some previous results for generic construction of a CBE from either existing IBE or CLE scheme, they are not comparable in efficiency to our scheme. When compare to the one proposed in [16], we enjoy a number of efficiency improvements:

1. We greatly reduce the size of public parameters. Their scheme requires the size of public parameters to be $n + 4$ group elements, where n is the length

of a bitstring representing the user public information (e.g. hash of public key). n should be at least 160 in order to claim a reasonable security. On the other side, we just need 5 group elements, no matter how large the user public information is.

2. Their scheme requires Boneh-Katz transform [6] to achieve CCA security. It needs a message authentication code (MAC) and an encapsulation scheme in addition to the basic scheme. One of the main drawback is the reduction of message space. Normally for a pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, usually the size of a group element in $\mathbb{G}_T$ representation is about 1024 bits. Without the Boneh-Katz transform, the message space of their scheme is $\mathbb{G}_T$, that is, 1024 bits. However, after applying the transform, it is reduced by at least 448 bits [6] due to the additional encapsulation information. Thus it only allows to encrypt a 576 bits message for a single encryption operation. In contrast, our scheme achieves CCA security directly without any transformation. Our message space remains 1024 bits. A larger message space implies that it requires a smaller number of encryption operations (which includes pairings and exponentiations) for the same plaintext, resulting in a smaller overall ciphertext as well.

Organization. In the rest of the paper, it is organized as follow. We review some preliminaries in Section 2. Security model is given in Section 3. Our proposed CBE scheme is presented in Section 4. Finally a concluding remarks is given in Section 5.

2 Preliminaries

2.1 Notations

Pairing. Let e be a bilinear map such that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ such that it has the following properties:

- $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic multiplicative groups of prime order p.
- each element of $\mathbb{G}$ and $\mathbb{G}_T$ has unique binary representation.
- g is a generator of $\mathbb{G}$.
- (Bilinear) $\forall x, y \in \mathbb{G}$ and a, $b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.
- (Non-degenerate) $e(g, g) \neq 1$.

2.2 Mathematical Assumptions

Definition 1 (Truncated Decision q-Augmented Bilinear Diffie-Hellman Exponent Assumption (q-ABDHE)). *We define the truncated decision q-ABDHE problem [11] as follows: Given a vector of $q + 3$ elements:*

$$\left(\tilde{g}, \tilde{g}^{(\alpha)^{q+2}}, g, g^{\alpha}, g^{(\alpha)^2}, \ldots, g^{(\alpha)^q}\right) \in \mathbb{G}^{q+3}$$

and an element $Z \in \mathbb{G}_T$ as input, output 1 if $Z = e(g^{(\alpha)^{q+1}}, \tilde{g})$ and output 0 otherwise. We say that the decision (t, ϵ, q)-ABDHE assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if no t-time algorithm has advantage at least ϵ over random guessing in solving the decision q-ABDHE problem in $(\mathbb{G}, \mathbb{G}_T)$.

Definition 2 (Decisional Bilinear Diffie-Hellman (DBDH) Assumption). *The Decisional Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}$ is defined as follows: On input $(g, g^a, g^b, g^c) \in \mathbb{G}^4$ and $Z \in \mathbb{G}_T$, output 1 if $Z = e(g,g)^{abc}$ and 0 otherwise. We say that the (t, ϵ)-DBDH assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if no t-time algorithm has advantage at least ϵ over random guessing in solving the DBDH problem in $(\mathbb{G}, \mathbb{G}_T)$.*

3 Security Model

We use the simplified model of [2] in the definition of our scheme and the security model.

Definition 3. *A certificate-based encryption (CBE) scheme is defined by six algorithms:*

- *Setup is a probabilistic algorithm taking as input a security parameter. It returns the certifier's master key msk and public parameters $param$. Usually this algorithm is run by the CA.*
- *UserKeyGen is a probabilistic algorithm that takes $param$ as input. When run by a client, it returns a public key PK and a secret key usk.*
- *Certify is a probabilistic algorithm that takes as input $(msk, \tau, param, \lambda, PK)$ where λ is a bit string containing user identification information. It returns $Cert'_\tau$ which is sent to the client. Here τ is a string identifying a time period.*
- *Consolidate is a deterministic certificate consolidation algorithm taking as input $(param, \tau, Cert'_\tau)$ and optionally $Cert_{\tau-1}$. It returns $Cert_\tau$, the certificate used by a client in time period τ.*
- *Encrypt is a probabilistic algorithm taking as input $(\tau, param, \lambda, PK, m)$ where m is a message. It outputs a ciphertext C.*
- *Decrypt is a deterministic algorithm taking $(param, Cert_\tau, usk, C)$ as input in time period τ. It returns either a message m or the special symbol $\perp$ indicating a decryption failure.*

We require that if C is the result of applying algorithm Encrypt with intput $(\tau, param, PK, m)$ and (usk, PK) is a valid key-pair, then m is the result of applying algorithm Decrypt on input $(param, Cert_\tau, usk, C)$, where $Cert_\tau$ is the output of Certify and Consolidate algorithms on input $(msk, param, \tau, PK)$. That is, we have

$$\mathsf{Decrypt}_{Cert_{\tau,usk}}(\mathsf{Encrypt}_{\tau,PK}(m)) = m$$

We also note that a concrete CBE scheme may not involve certificate consolidation. In this situation, algorithm Consolidate will simply output $Cert_\tau = Cert'_\tau$.

In the rest of this paper, for simplicity, we will omit Consolidate and the time identifying string τ in all notations.

The security of CBE is defined by two different games and the adversary chooses which game to play. In Game 1, the adversary models an uncertified entity while in Game 2, the adversary models the certifier in possession of the master key msk attacking a fixed entity's public key.

Definition 4 (CBE Game 1). *The challenger runs* Setup*, gives param to the adversary* $\mathcal{A}_1$ *and keeps* msk *to itself. The adversary then interleaves certification and decryption queries with a single challenge query. These queries are answered as follows:*

- *On certification query* (λ, PK, usk)*, the challenger checks that* (PK, usk) *is a valid key-pair. If so, it runs* Certify *on input* $(msk, param, \lambda, PK)$ *and returns* $Cert$*. Else it returns* $\perp$*.*
- *On decryption query* (λ, PK, usk, C)*, the challenger checks that* (PK, usk) *is a valid key-pair. If so, it generates* $Cert$ *by using algorithm* Certify *with inputs* $(msk, param, \lambda, PK)$ *and outputs* $\mathsf{Decrypt}_{Cert,usk}(C)$*. Else it returns* $\perp$*.*[1]
- *On challenge query* $(\lambda^*, PK^*, usk^*, m_0, m_1)$*, the challenger checks that* (PK^*, usk^*) *is a valid key-pair. If so, it chooses a random bit* $b \in_R \{0, 1\}$ *and returns* $C^* = \mathsf{Encrypt}_{\lambda^*, PK^*}(m_b)$*. Else it returns* $\perp$*.*

Finally $\mathcal{A}_1$ *outputs a bit* $b' \in \{0, 1\}$*. The adversary wins the game if* $b = b'$ *and* $(\lambda^*, PK^*, usk^*, C^*)$ *was not submitted to the decryption oracle after the challenge, and* (λ^*, PK^*, usk^*) *was not submitted to the certification query. We define* $\mathcal{A}_1$*'s advantage in this game to be* $Adv(\mathcal{A}_1) = 2|\Pr[b = b'] - \frac{1}{2}|$*.*

Definition 5 (CBE Game 2). *The challenger runs* Setup*, gives param and* msk *to the adversary* $\mathcal{A}_2$*. The challenger then runs* UserKeyGen *to obtain a key-pair* (PK^*, usk^*) *and gives* λ^*, PK^* *to the adversary* $\mathcal{A}_2$*. The adversary interleaves decryption queries with a single challenge query. These queries are answered as follows:*

- *On decryption query* (C)*, the challenger generates* $Cert$*, by using algorithm* Certify *with inputs* $(msk, param, \lambda^*, PK^*)$*. It then outputs* $\mathsf{Decrypt}_{Cert,usk^*}(C)$*.*
- *On challenge query* (m_0, m_1)*, the challenger randomly chooses a bit* $b \in_R \{0, 1\}$ *and returns* $C^* = \mathsf{Encrypt}_{\lambda^*, PK^*}(m_b)$*.*

Finally $\mathcal{A}_2$ *outputs a guess* $b' \in \{0, 1\}$*. The adversary wins the game if* $b = b'$ *and* C^* *was not submitted to the decryption oracle after the challenge. We define* $\mathcal{A}_2$*'s advantage in this game to be* $Adv(\mathcal{A}_2) = 2|\Pr[b = b'] - \frac{1}{2}|$*.*

We note that our model does not support security against *Malicious Certifier*. That is, we assume that the certifier generates all public parameters honest, according to the algorithm specified. The adversarial certifier is only given the master secret key, instead of allowing to generate all public parameters. Although malicious certifier has not been discussed in the literature, similar concept of *Malicious Key Generation Centre (KGC)* [3] has been formalized in the area of certificateless cryptography.

Definition 6 (Secure CBE). *A CBE scheme is said to be* (t, q_c, q_d, ϵ)*-secure against adaptive chosen ciphertext attack if all* t*-time adversary making at most*

[1] Note that in the decryption oracle of Game 1, we need to take the user secret key as input. This is the same as all previous CBE schemes [10,2].

q_c certification query (for Game 1 only) and at most q_d chosen ciphertext decryption queries have advantage at most ϵ in either CBE Game 1 or CBE Game 2.

4 The Proposed Scheme

4.1 Construction

Our scheme is motivated by Gentry's identity based encryption scheme [11]. Details are as follow.

Setup. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a pairing. Let p be the group order of $\mathbb{G}$ and $\mathbb{G}_T$. The CA chooses generators $g, g_2, g_2', g_2'' \in \mathbb{G}$ and randomly selects $\alpha \in_R \mathbb{Z}_p$. It computes $g_1 = g^\alpha$. It also chooses two hash functions $H, H' : \{0,1\}^* \rightarrow \mathbb{Z}_p$ from a family of universal one-way hash functions. The master secret msk is α while the public parameters $param$ are $(H, H', e, p, g, g_1, g_2, g_2', g_2'')$.

UserKeyGen. The user randomly selects $\beta, \gamma, \delta, \xi, \delta', \xi' \in_R \mathbb{Z}_p$ and computes $h_1 = g^\beta$, $h_2 = g^\gamma$, $h_3 = g^\delta$, $h_4 = g^\xi$, $h_3' = g^{\delta'}$, $h_4' = g^{\xi'}$. The public key PK is $(h_1, h_2, h_3, h_4, h_3', h_4')$ and the user secret key usk is $(\beta, \gamma, \delta, \xi, \delta', \xi')$.

Certify. Suppose a user with public key PK and identification information $\lambda \in \{0,1\}^*$ wants to be certified. He sends $pk = (h_1, h_2, h_3, h_4, h_3', h_4')$ and λ to the CA. The CA randomly selects $r_1, r_1', r_1'' \in_R \mathbb{Z}_p$ and computes $h = H(h_1, h_2, h_3, h_4, h_3', h_4', \lambda)$ and

$$r_2 = (g_2 g^{-r_1})^{\frac{1}{\alpha - h}} \qquad r_2' = (g_2' g^{-r_1'})^{\frac{1}{\alpha - h}} \qquad r_2'' = (g_2'' g^{-r_1''})^{\frac{1}{\alpha - h}}$$

The certificate $Cert$ is $(r_1, r_2, r_1', r_2', r_1'', r_2'')$. Similar to [11], we require that the CA always uses the same random values r_1, r_1', r_1'' for this user. This can be accomplished, for example, by using an internal log to ensure consistency.

Encrypt. To encrypt a message $m \in \mathbb{G}_T$ using public key $(h_1, h_2, h_3, h_4, h_3', h_4')$ and λ, randomly selects $s \in_R \mathbb{Z}_p$, computes $h = H(h_1, h_2, h_3, h_4, h_3', h_4', \lambda)$ and

$$C_1 = g_1^s g^{-sh} \qquad C_2 = e(g,g)^s \qquad C_3 = m \cdot e(g, g_2)^{-s} \cdot e(h_1, h_2)^{-s}$$

$$C_4 = e(g, g_2')^s \cdot e(g, g_2'')^{s\phi} \cdot e(h_1, h_2)^s \cdot e(h_3, h_4)^s \cdot e(h_3', h_4')^{s\phi}$$

where $\phi = H'(C_1, C_2, C_3)$. Outputs the ciphertext $C = (C_1, C_2, C_3, C_4)$.

Decrypt. To decrypt ciphertext $C = (C_1, C_2, C_3, C4)$ with certificate $(r_1, r_2, r_1', r_2', r_1'', r_2'')$ and secret key $(\beta, \gamma, \delta, \xi, \delta', \xi')$ with respect to public key $(h_1, h_2, h_3, h_4, h_3', h_4')$ and λ, computes

$$m = C_3 \cdot e(C_1, r_2) \cdot (C_2)^{r_1 + \beta\gamma}$$

and $\phi = H'(C_1, C_2, C_3)$. Outputs m if

$$C_4 = e(C_1, r_2' r_2''^{\phi})(C_2)^{r_1' + r_1''\phi + \beta\gamma + \delta\xi + \delta'\xi'\phi}$$

Otherwise outputs $\perp$.

Correctness. If the ciphertext is well formed, we have

$$\begin{aligned}
& e(C_1, r'_2 r''^{\phi}_2)(C_2)^{r'_1 + r''_1\phi + \beta\gamma + \delta\xi + \delta'\xi'\phi} \\
&= e\big(g_1^s g^{-sh}, (g'_2 g^{-r'_1})^{\frac{1}{\alpha - h}} (g''_2 g^{-r''_1})^{\frac{\phi}{\alpha - h}}\big) \\
&\quad \cdot e(g,g)^{s(r'_1 + r''_2\phi)} \cdot e(g,g)^{s\beta\gamma} \cdot e(g,g)^{s\delta\xi} \cdot e(g,g)^{s\delta'\xi'\phi} \\
&= e(g^s, g'_2 g''^{\phi}_2) \cdot e(g^s, g^{-(r'_1 + r''_1\phi)}) \cdot \ e(g,g)^{s(r'_1 + r''_2\phi)} \\
&\quad \cdot e(g^\beta, g^\gamma)^s \cdot e(g^\delta, g^\xi)^s \cdot e(g^{\delta'}, g^{\xi'})^{s\phi} \\
&= e(g, g'_2)^s \cdot e(g, g''_2)^{s\phi} \cdot e(h_1, h_2)^s \cdot e(h_3, h_4)^s \cdot e(h'_3, h'_4)^{s\phi} \\
&= C_4
\end{aligned}$$

On the other side, we have

$$\begin{aligned}
& C_3 \cdot e(C_1, r_2) \cdot (C_2)^{r_1 + \beta\gamma} \\
&= m \cdot e(g,g_2)^{-s} \cdot e(h_1, h_2)^{-s} \cdot e\big(g_1^s g^{-sh}, (g_2 g^{-r_1})^{\frac{1}{\alpha - h}}\big) \\
&\quad \cdot e(g,g)^{sr_1} \cdot e(g,g)^{s\beta\gamma} \\
&= m \cdot e(g,g_2)^{-s} \cdot e(h_1, h_2)^{-s} \cdot e\big(g^{s(\alpha - h)}, (g_2 g^{-r_1})^{\frac{1}{\alpha - h}}\big) \\
&\quad \cdot e(g,g)^{sr_1} \cdot e(g^\beta, g^\gamma)^s \\
&= m \cdot e(g,g_2)^{-s} \cdot e(h_1, h_2)^{-s} \cdot e(g,g_2)^s \cdot e(g,g)^{-sr_1} \cdot \cdot e(g,g)^{sr_1} \cdot e(h_1, h_2)^s \\
&= m.
\end{aligned}$$

4.2 Security Analysis

Theorem 1. *Let $q = q_c + 1$ where q_c is the number of certification query allowed. Assume thetruncated decision (t, ϵ, q)-ABDHE assumption holds for $(\mathbb{G}, \mathbb{G}_T, e)$. Then our proposed CBE scheme is $(t', \epsilon', q_c, q_d)$ secure against Game 1 adversary, where*

$$t' = t - \mathcal{O}(t_{exp} \cdot q^2) \qquad \epsilon' = \epsilon + q_d/p$$

where t_{exp} is the time required for an exponentiation in $\mathbb{G}$.

Proof. The Game 1 security of our scheme is more or less similar to the IND-ID-CCA security of Gentry's IBE scheme [11]. In this extended abstract, we may omit some of the details here. Readers may refer to [11] for the full explanation of some steps.

Let $\mathcal{A}_1$ be an adversary that $(t', \epsilon', q_c, q_d)$-wins Game 1. We construct an algorithm $\mathcal{B}$ that solves the truncated decision q-ABDHE problem, as follows. $\mathcal{B}$ takes an input a random truncated decision q-ABDHE challenge $(\tilde{g}, \tilde{g}_{(q+2)}, g, g_{(1)}, \ldots, g_{(q)}, Z)$, where Z is either $e(g_{(q+1)}, \tilde{g})$ or a random element of $\mathbb{G}_T$ (we denote $g_{(i)} = g^{(\alpha^i)}$). Algorithm $\mathcal{B}$ proceeds as follow.

Setup: $\mathcal{B}$ generates random polynomials $f(x), f'(x), f''(x) \in \mathbb{Z}_p[x]$ of degree q. It sets $g_1 = g_{(1)}$ and $g_2 = g^{f(\alpha)}$, computing g_2 from $(g, g_{(1)}, \ldots, g_{(q)})$. Similarly,

it also sets $g_2' = g^{f'(\alpha)}$ and $g_2'' = g^{f''(\alpha)}$. $\mathcal{B}$ also chooses two hash functions $H, H' : \{0,1\}^* \to \mathbb{Z}_p$ from a family of universal one-way hash functions.

It sends the public parameters $(H, H', g, g_1, g_2, g_2', g_2'')$ to $\mathcal{A}_1$. Since $g, \alpha, f(x)$, $f'(x), f''(x)$ are chosen uniformly at random, g_2, g_2', g_2'' are also uniformly random and the public parameters have a distribution identical to that in the actual construction.

Oracle Queries:

- *Certification Query*: $\mathcal{B}$ responds to a query on public key $PK = (h_1, h_2, h_3, h_4, h_3', h_4')$, user identification information λ and secret key $usk = (\beta, \gamma, \delta, \xi, \delta', \xi')$. $\mathcal{B}$ checks whether PK is corresponding to usk. If it is not, output $\perp$. Then it computes $h = H(h_1, h_2, h_3, h_4, h_3', h_4', \lambda)$. If $h = \alpha$, $\mathcal{B}$ uses α to solve truncated decision q-ABDHE immediately. Else, to generate let $F_h(x)$ denote the $(q-1)$-degree polynomial $(f(x) - f(h))/(x - h)$. $\mathcal{B}$ sets $(r_1, r_2) = (f(h), g^{F_h(\alpha)})$. These are valid certificate values for h, since

$$g^{F_h(\alpha)} = g^{(f(\alpha)-f(h))/(\alpha-h)} = (g_2 g^{-f(h)})^{1/(\alpha-h)}$$

  as required. It computes the remainder of the certificate in a similar way.
- *Decryption Query*: To respond to a decryption query on (λ, PK, usk, C), $\mathcal{B}$ generates a certificate for PK as above. It then decrypts C by performing the usual Decrypt algorithm with the certificate and the secret key usk.

Challenge: $\mathcal{A}_1$ outputs a challenged public key $PK^* = (h_1^*, h_2^*, h_3^*, h_4^*, h'^*_3, h'^*_4)$, user identification information λ^*, secret key $usk^* = (\beta^*, \gamma^*, \delta^*, \xi^*, \delta'^*, \xi'^*)$ and two messages m_0, m_1. Again, $\mathcal{B}$ checks whether PK^*, usk^* is a valid key pair. It outputs $\perp$ if it is not. Else, $\mathcal{B}$ computes $h^* = H(h_1^*, h_2^*, h_3^*, h_4^*, h'^*_3, h'^*_4, \lambda^*)$. If $h^* = \alpha$, $\mathcal{B}$ uses α to solve truncated decision q-ABDHE immediately. Else $\mathcal{B}$ chooses a random bit $b \in_R \{0,1\}$, and computes a certificate $(r_1, r_2, r_1', r_2', r_1'', r_2'')$ for PK^* as in the certification query. Let $f_2(x) = x^{q+2}$ and let $F_{2,h^*}(x) = (f_2(x) - f_2(h^*))/\ (x - h^*)$, which is a polynomial of degree $q+1$. $\mathcal{B}$ sets

$$C_1^* = \tilde{g}^{f_2(\alpha)-f_2(h^*)}\ ;\ C_2^* = Z \cdot e(\tilde{g}, \prod_{i=0}^{q} g^{F_{2,h^*,i}\alpha^i})\ ;\ C_3^* = \frac{m_b}{e(C_1^*, r_2)(C_2^*)^{r_1}(C_2^*)^{\beta^*\gamma^*}}$$

where $F_{2,h^*,i}$ is the coefficient of x^i in $F_{2,h^*}(x)$. After setting $\phi = H'(C_1^*, C_2^*, C_3^*)$, $\mathcal{B}$ sets

$$C_4^* = e(C_1^*, r_2' r_2''^{\phi})(C_2^*)^{r_1' + r_1''\phi + \beta^*\gamma^* + \delta^*\xi^* + \delta'^*\xi'^*\phi}$$

It sends $(C_1^*, C_2^*, C_3^*, C_4^*)$ to $\mathcal{A}_1$ as the challenge ciphertext.

Let $s = (\log_g \tilde{g}) F_{2,h^*}(\alpha)$. If $Z = e(g_{(q+1)}, \tilde{g})$, then $C_1^* = g^{s(\alpha - h^*)}$ and

$$m_b / C_3^* = e(C_1^*, r_2)(C_2^*)^{r_1}(C_2^*)^{\beta^*\gamma^*} = e(g, g_2)^s e(h_1^*, h_2^*)^s$$

Thus $(C_1^*, C_2^*, C_3^*, C_4^*)$ is a valid ciphertext for (PK^*, m_b) under randomness s. Since $\log_g \tilde{g}$ is uniformly random, s is random, and so $(C_1^*, C_2^*, C_3^*, C_4^*)$ is a valid, appropriately-distributed challenge to $\mathcal{A}_1$.

Output: Finally $\mathcal{A}_1$ outputs a bit $b' \in \{0,1\}$. If $b = b'$, $\mathcal{B}$ outputs 1 indicating that $Z = e(g_{(q+1)}, \tilde{g})$. Otherwise it outputs 0.

Probability Analysis: When $\mathcal{B}$'s input is sampled according to the problem instance, $\mathcal{B}$'s simulation appears perfect to $\mathcal{A}_1$ if $\mathcal{A}_1$ makes only certification queries. $\mathcal{B}$'s simulation still appears perfect if $\mathcal{A}_1$ makes decryption queries only on public keys for which it queries the certificate, since $\mathcal{B}$'s responses give $\mathcal{A}_1$ no additional information. Furthermore, querying well-formed ciphertexts to the decryption oracle does not help $\mathcal{A}_1$ distinguish between the simulation and the actual construction, since by the correctness of Decrypt, well-formed ciphertexts will be accepted. Finally querying a non-well-formed ciphertext does not help $\mathcal{A}_1$ distinguish, since this ciphertext will fail the Decrypt check under every valid certificate. Thus, by following the approach of Lemma 1 of [11], we claim that the decryption oracle, in the simulation and in the actual construction, rejects all invalid ciphertexts under public keys not queried by $\mathcal{A}_1$, except with probability q_d/p.

Time Complexity: In the simulation, $\mathcal{B}$'s overhead is dominated by computing $g^{F_h(\alpha)}$ in response to $\mathcal{A}_1$'s certification query on PK, where $F_h(x)$ is a polynomial of degree $q-1$. Each such computation requires $\mathcal{O}(q)$ exponentiations in $\mathbb{G}$. Since $\mathcal{A}_1$ makes at most $q-1$ such queries, $t = t' + \mathcal{O}(t_{exp} \cdot q^2)$. □

Theorem 2. *Assume (t,ϵ)-DBDH assumption holds for $(\mathbb{G}, \mathbb{G}_T, e)$. Then our proposed CBE scheme is $(t', \epsilon', q_c, q_d)$ secure against Game 2 adversary, where*

$$t' = t \qquad \epsilon' = \epsilon + q_d/p$$

Proof. Let $\mathcal{A}_2$ be an adversary that $(t', \epsilon', q_c, q_d)$-wins Game 2. We construct an algorithm $\mathcal{B}$ that solves the DBDH problem, as follows. $\mathcal{B}$ takes an input a random DBDH challenge (g, g^a, g^b, g^c, Z), where Z is either $e(g,g)^{abc}$ or a random element of $\mathbb{G}_T$. Algorithm $\mathcal{B}$ proceeds as follow.

Setup: $\mathcal{B}$ randomly generates $\alpha, v, x, y, t, w, t', w' \in_R \mathbb{Z}_p$ and sets $g_1 = g^\alpha$, $g_2 = g^v$, $g_2' = g^x$, $g_2'' = g^y$. $\mathcal{B}$ also chooses two hash functions $H, H' : \{0,1\}^* \rightarrow \mathbb{Z}_p$ from a family of universal one-way hash functions.

The public parameters *param* are $(H, H', g, g_1, g_2, g_2', g_2'')$ while the master secret key msk is α. $\mathcal{B}$ also sets $h_1 = g^a$, $h_2 = g^b$, $h_3 = g^t$, $h_4 = g^w$, $h_3' = g^{t'}$, $h_4' = g^{w'}$ as the challenged public key PK. $\mathcal{B}$ also constructs some binary string as the identification information λ. $(param, msk, \lambda, PK)$ are given to the adversary $\mathcal{A}_2$. Obviously the public parameters and the challenged public key have a distribution identical to that in the actual construction.

Oracle Queries:

– *Decryption Query:* To respond to a decryption query on $C = (C_1, C_2, C_3, C_4)$, $\mathcal{B}$ first computes $\phi = H'(C_1, C_2, C_3)$ and

$$\begin{aligned}
&\frac{C_4 \cdot C_3 \cdot C_2^v}{C_2^{x+y\phi+tw+t'w'\phi}} \\
&= \frac{e(g,g_2')^s \cdot e(g,g_2'')^{s\phi} \cdot e(h_1,h_2)^s \cdot e(h_3,h_4)^s \cdot e(h_3',h_4')^{s\phi}}{e(g,g^x)^s \cdot e(g,g^y)^{s\phi} \cdot e(g^t,g^w)^s \cdot e(g^{t'},g^{w'})^{s\phi}} \\
&\quad \cdot m \cdot e(g,g_2)^{-s} \cdot e(h_1,h_2)^{-s}\ e(g,g^v)^s \\
&= m
\end{aligned}$$

It then generates the certificate $(r_1, r_2, r_1', r_2', r_1'', r_2'')$ using the knowledge of α, and further checks

$$C_4 \stackrel{?}{=} \frac{m \cdot e(C_1, r_2' r_2''^{\phi}) \cdot (C_2)^{r_1'+r_1''\phi+tw+t'w'\phi-v}}{C_3}$$

It outputs m if it is equal, otherwise outputs $\perp$.

Challenge: $\mathcal{A}_2$ outputs two messages m_0, m_1. $\mathcal{B}$ randomly chooses a bit $b \in_R \{0,1\}$, computes $h = H(h_1, h_2, h_3, h_4, h_3', h_4', \lambda)$ and sets

$$C_1^* = (g^c)^{\alpha-h} \qquad C_2^* = e(g,g^c) \qquad C_3 = m_b \cdot e(g^c, g^{-v}) \cdot Z^{-1}$$

$$C_4 = e(g^c, g^x) \cdot e(g^c, g^v)^{\phi} \cdot Z \cdot e(g^c, g^{tw}) \cdot e(g^c, g^{t'w'})^{\phi}$$

where $\phi = H'(C_1^*, C_2^*, C_3^*)$. The challenged ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ is sent to $\mathcal{A}$. It can be easily seen that if $Z = e(g,g)^{abc}$, C^* is a valid, appropriately-distributed challenge ciphertext.

Output: Finally $\mathcal{A}_2$ outputs a bit $b' \in \{0,1\}$. If $b = b'$, $\mathcal{B}$ outputs 1 indicating that $Z = e(g,g)^{abc}$. Otherwise it outputs 0.

Probability Analysis: Similar to Game 1, the simulation remains perfect except with probability q_d/p that the decryption oracle will not reject all invalid ciphertext.

Time Complexity: The time complexity for $\mathcal{B}$ depends only on $\mathcal{A}$. Thus we have $t' = t$. □

4.3 Efficiency Analysis

Previous generic constructions [18,7,2] are not comparable to our scheme in terms of efficiency. When compare to the one in [16] (the only concrete implementation that is fully secure in the standard model), we enjoy a great improvement in terms of space efficiency. First, our scheme requires just 5 group elements (about 800 bits, assuming each group element costs 160 bits in the optimal case) in the public parameters. But the scheme in [16] needs 164 group elements (about 26240 bits) in order to achieve the same level of security.

Second, the message space of our scheme is in $\mathbb{G}_T$, which is about 1024 bits. The message space of the scheme in [16] is around 576 bits only. The main

reason for this difference is that. Although the chosen plaintext secure (CPA) version of the scheme in [16] allows the message space to be in $\mathbb{G}_T$, in order achieve CCA security, it requires an additional encapsulation scheme and the Boneh-Katz transform. The transform modifies the scheme a bit, by encrypting $M = m||dec$ where m is the original message, dec is the decommitment string and M is the combined message which should be in $\mathbb{G}_T$. According to [6], the suggested length of dec should be at least 448 bits. That is, the message space of the original message is reduced to $1024 - 448 = 576$ bits. If we want to encrypt a message of 1024 bits, we need to split it into two parts and encrypt it part by part. It results in a double increase of both computation cost and ciphertext size, and maybe security reduction as well. This difference becomes significant if we want to encrypt a large message. On the other side, as we do not require any transformation or encapsulation scheme to achieve CCA security, our message space can be remained as 1024 bits, without suffering any efficiency reduction.

In terms of computation cost, although we require some pairing operations in the encryption algorithm, they can be pre-computed by the CA and given as part of the public parameters. For those pairing operations related to the public key of the intended receiver, they can be pre-computed by the receiver and given as part of the public key as well. In this way, the encryptor does not need to compute any pairing operation. For the decryption algorithm, we require two pairing operations.

5 Concluding Remarks

In this paper, we propose a new CBE scheme which is motivated from Gentry's IBE scheme [11]. Our scheme is fully CCA secure in the standard model. We do not require any MAC or encapsulation scheme to achieve CCA security. This facilitates us to achieve significant improvement in efficiency when compared to [16]. We believe the concept of certificate-based encryption with our efficient implementation allows it to be used in some practical applications, and particularly suitable to be employed in computation limited devices, or wireless sensor network.

We also remark that our scheme does not support malicious CA security. That is, we assume that the CA generates the public parameters according to the algorithm honestly. This is the same as all CBE schemes in the literature. However, recently Au *et al.* [3] pointed out that a malicious KGC in certificateless cryptography (with respect to CA in certificate-based setting) may pose some security risks to the system, by generating the public parameters in a malicious way. We note that our system (and all previous CBE schemes) may suffer similar attack. We have not discussed those risks in this paper. We leave it as an open problem to future research.

References

1. Al-Riyami, S.S., Paterson, K.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Al-Riyami, S.S., Paterson, K.G.: CBE from CL-PKE: A generic construction and efficient schemes. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 398–415. Springer, Heidelberg (2005)
3. Au, M., Chen, J., Liu, J., Mu, Y., Wong, D., Yang, G.: Malicious KGC attacks in certificateless cryptography. In: ASIACCS 2007, pp. 302–311. ACM Press, New York (2007)
4. Au, M., Liu, J., Susilo, W., Yuen, T.: Certificate based (linkable) ring signature. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 79–92. Springer, Heidelberg (2007)
5. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Katz, J.: Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
7. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005)
8. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
9. Galindo, D., Morillo, P., Ràfols, C.: Breaking Yum and Lee generic constructions of certificate-less and certificate-based encryption schemes. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 81–91. Springer, Heidelberg (2006)
10. Gentry, C.: Certificate-based encryption and the certificate revocation problem. In: EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
11. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
12. Kang, B.G., Park, J.H., Hahn, S.G.: A certificate-based signature scheme. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 99–111. Springer, Heidelberg (2004)
13. Li, J., Huang, X., Mu, Y., Susilo, W., Wu, Q.: Certificate-based signature: Security model and efficient construction. In: López, J., Samarati, P., Ferrer, J.L. (eds.) EuroPKI 2007. LNCS, vol. 4582, pp. 110–125. Springer, Heidelberg (2007)
14. Liu, J., Au, M., Susilo, W.: Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model. In: ASIACCS 2007, pp. 273–283. ACM Press, New York (2007)
15. Liu, J., Baek, J., Susilo, W., Zhou, J.: Certificate based signature schemes without pairings or random oracles. In: ISC 2008. LNCS, vol. 5222. Springer, Heidelberg (to appear, 2008)
16. Morillo, P., Ràfols, C.: Certificate-based encryption without random oracles (2006), `http://eprint.iacr.org/2006/012/`
17. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
18. Yum, D.H., Lee, P.J.: Identity-based cryptography in public key management. In: Katsikas, S.K., Gritzalis, S., López, J. (eds.) EuroPKI 2004. LNCS, vol. 3093, pp. 71–84. Springer, Heidelberg (2004)

An Improved Robust Fuzzy Extractor

Bhavana Kanukurthi and Leonid Reyzin

Boston University Computer Science
111 Cummington St., Boston, MA 02215, USA
http://cs-people.bu.edu/bhavanak, http://www.cs.bu.edu/~reyzin

Abstract. We consider the problem of building robust fuzzy extractors, which allow two parties holding similar random variables W, W' to agree on a secret key R in the presence of an active adversary. Robust fuzzy extractors were defined by Dodis et al. in Crypto 2006 to be noninteractive, i.e., only one message P, which can be modified by an unbounded adversary, can pass from one party to the other. This allows them to be used by a single party at different points in time (e.g., for key recovery or biometric authentication), but also presents an additional challenge: what if R is used, and thus possibly observed by the adversary, before the adversary has a chance to modify P. Fuzzy extractors secure against such a strong attack are called post-application robust.

We construct a fuzzy extractor with post-application robustness that extracts a shared secret key of up to $(2m-n)/2$ bits (depending on error-tolerance and security parameters), where n is the bit-length and m is the entropy of W. The previously best known result, also of Dodis et al., extracted up to $(2m-n)/3$ bits (depending on the same parameters).

1 Introduction

Consider the following scenario. A user Charlie has a secret w that he wants to use to encrypt and authenticate his hard drive. However, w is not a uniformly random key; rather, it is a string with some amount of entropy from the point of view of any adversary $\mathcal{A}$. Naturally, Charlie uses an extractor [NZ96], which is a tool for converting entropic strings into uniform ones. An extractor Ext is an algorithm that takes the entropic string w and a uniformly random seed i, and computes $R = \mathsf{Ext}(w; i)$ that is (almost) uniformly random even given i.

It may be problematic for Charlie to memorize or store the uniformly random R (this is in contrast to w, which can be, for example, a long passphrase already known to Charlie, his biometric, or a physical token, such as a physical one-way function [PRTG02]). Rather, in order to decrypt the hard drive, Charlie can use i again to recompute $R = \mathsf{Ext}(w; i)$. The advantage of storing i rather than R is that i need not be secret, and thus can be written, for example, on an unencrypted portion of the hard drive.

Even though the storage of i need not be secret, the authenticity of i is very important. If $\mathcal{A}$ could modify i to i', then Charlie would extract some related key R', and any guarantee on the integrity of the hard drive would vanish, because typical encryption and authentication schemes do not provide

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 156–171, 2008.

any security guarantees under related-key attacks. To authenticate i, Charlie would need to use some secret key, but the only secret he has is w.

This brings us to the problem of building *robust* extractors: ones in which the authenticity of the seed can be verified at reconstruction time. A robust extractor has two procedures: a randomized $\mathsf{Gen}(w)$, which generates (R, P) such that R is uniform even given P (think of P as containing the seed i as well as some authentication information), and $\mathsf{Rep}(w, P')$, which reproduces R if $P' = P$ and outputs $\perp$ with high probability for an adversarially produced $P' \neq P$.

Note that in the above scenario, the adversary $\mathcal{A}$, before attempting to produce $P' \neq P$, gets to see the value P and how the value R is used for encryption and authentication. Because we want robust fuzzy extractors to be secure for a wide variety of applications, we do not wish to restrict how R is used and, therefore, what information about R is available to $\mathcal{A}$. Rather, we will require that $\mathcal{A}$ has low probability of getting $\mathsf{Rep}(w, P')$ to not output $\perp$ *even* if $\mathcal{A}$ is given both P and R. This strong notion of security is known as *post-application* robustness.

An additional challenge may be that the value w when Gen is run is slightly different from the value w' available when Rep is run: for example, the user may make a typo in a long passphrase, or a biometric reading may differ slightly. Extractors that can tolerate such differences and still reproduce R exactly are called *fuzzy* [DORS08]. Fuzzy extractors are obtained by adding error-correcting information to P, to enable Rep to compensate for errors in w'. The specific constructions depend on the kinds of errors that can occur (e.g., Hamming errors, edit distance errors, etc.).

Robust (fuzzy) extractors are useful not only in the single-party setting described above, but also in interactive settings, where two parties are trying to derive a key from a shared (slightly different in the fuzzy case) secret w that either is nonuniform or about which some limited information is known to the adversary $\mathcal{A}$. One party, Alice, can run Gen to obtain (R, P) and send P to the other party, Bob, who can run Rep to also obtain R. However, if $\mathcal{A}$ is actively interfering with the channel between Alice and Bob and modifying P, it is important to ensure that Bob detects the modification rather than derives a different key R'. Moreover, unless Alice can be sure that Bob truly received P before she starts using R in a communication, post-application robustness is needed.

Prior Work. Fuzzy extractors, defined in [DORS08], are essentially the noninteractive variant of privacy amplification and information reconciliation protocols, considered in multiple works, including [Wyn75, BBR88, Mau93, BBCM95]. Robust (fuzzy) extractors, defined in [BDK+05, DKRS06], are the noninteractive variant of privacy amplification (and information reconciliation) secure against active adversaries [Mau97, MW97, Wol98, MW03, RW03, RW04].

Let the length of w be n and the entropy of w be m. Post-application robust fuzzy extractors cannot extract anything out of w if $m < n/2$, because an extractor with post-application robustness implies an information-theoretically secure message authentication code (MAC) with w as the key[1], which is impossible if

[1] The MAC is obtained by extracting R, using it as a key to any standard information-theoretic MAC (e.g., [WC81]), and sending P along with the tag to the verifier.

$m < n/2$ (see [DS02] for impossibility of deterministic MACs if $m < n/2$ and its extension by [Wic08] to randomized MACs). Without any set-up assumptions, the only previously known post-application robust extractor, due to [DKRS06], extracts R of length $\frac{2}{3}(m - n/2 - \log\frac{1}{\delta})$ (or even less if R is required to be very close to uniform), where δ is the probability that the adversary violates robustness. Making it fuzzy further reduces the length of R by an amount related to the error-tolerance. (With set-up assumptions, one can do much better: the construction of [CDF+08] extracts almost the entire entropy m, reduced by an amount related to security and, in the fuzzy case, to error-tolerance. However, this construction assumes that a nonsecret uniformly random string is already known to both parties, and that the distribution on w, including adversarial knowledge about w, is independent of this string.)

Our Results. The robust extractor construction of [DKRS06] is parameterized by a value v that can be decreased in order to obtain a longer R. In fact, as shown in [DKRS06], a smaller v can be used for *pre-application* robustness (a weaker security notion, in which $\mathcal{A}$ gets P but not R). We show in Theorem 2 that the post-application-robustness analysis of [DKRS06] is essentially tight, and if v is decreased, the construction becomes insecure.

Instead, in Section 3, we propose a new construction of an extractor with post-application robustness that extracts R of length $m - n/2 - \log\frac{1}{\delta}$, improving the previous result by a factor of 3/2 (more if R is required to be very close to uniform). While this is only a constant-factor increase, in scenarios where secret randomness is scarce it can make a crucial difference. Like [DKRS06], we make no additional set-up assumptions. Computationally, our construction is slightly more efficient than the construction of [DKRS06]. Our improved robust extractor translates into an improved robust fuzzy extractor using the techniques of [DKRS06], with the same factor of 3/2 improvement.

In addition, we show (in Section 3.2) a slight improvement for the pre-application robust version of the extractor of [DKRS06], applicable when the extracted string must be particularly close to uniform.

2 Preliminaries

Notation. For binary strings a, b, $a||b$ denotes their concatenation, $|a|$ denotes the length of a. For a binary string a, for we denote by $[a]_i^j$, the substring $b = a_i a_{i+1} \dots a_j$. If S is a set, $x \leftarrow S$ means that x is chosen uniformly from S. If X is a probability distribution (or a random variable), then $x \leftarrow X$ means that x is chosen according to distribution X. If X and Y are two random variables, then $X \times Y$ denotes the product distribution (obtained by sampling X and Y *independently*). All logarithms are base 2.

Random Variables, Entropy, Extractors. Let U_l denote the uniform distribution on $\{0,1\}^l$. Let X_1, X_2 be two probability distributions over some set S. Their *statistical distance* is

$$\mathbf{SD}(X_1, X_2) \stackrel{\text{def}}{=} \max_{T \subseteq S} \{\Pr[X_1 \in T] - \Pr[X_2 \in T]\} = \frac{1}{2} \sum_{s \in S} \left| \Pr_{X_1}[s] - \Pr_{X_2}[s] \right|$$

(they are said to be ε-close if $\mathbf{SD}(X_1, X_2) \leq \varepsilon$). We will use the following lemma on statistical distance that was proven in [DKRS08]:

Lemma 1. *For any joint distribution (A, B) and distributions C and D over the ranges of A and B respectively, if $\mathbf{SD}((A, B), C \times D) \leq \alpha$, then $\mathbf{SD}((A, B), C \times B) \leq 2\alpha$.*

Min-entropy. The *min-entropy* of a random variable W is defined as $\mathbf{H}_\infty(W) = -\log(\max_w \Pr[W = w])$ (all logarithms are base 2, unless specified otherwise). Following [DORS08], for a joint distribution (W, E), define the (average) conditional min-entropy of W given E as

$$\widetilde{\mathbf{H}}_\infty(W \mid E) = -\log(\mathop{\mathbf{E}}_{e \leftarrow E}(2^{-\mathbf{H}_\infty(W|E=e)}))$$

(here the expectation is taken over e for which $Pr[E = e]$ is nonzero). A computationally unbounded adversary who receives the value of E cannot find the correct value of W with probability greater than $2^{-\widetilde{\mathbf{H}}_\infty(W|E)}$. We will use the following lemma from [DORS08]:

Lemma 2. *Let A, B, C be random variables. If B has at most 2^λ possible values, then $\widetilde{\mathbf{H}}_\infty(A|B, C) \geq \widetilde{\mathbf{H}}_\infty((A, B)|C) - \lambda \geq \widetilde{\mathbf{H}}_\infty(A|C) - \lambda$. In particular, $\widetilde{\mathbf{H}}_\infty(A|B) \geq \mathbf{H}_\infty((A, B)) - \lambda \geq \mathbf{H}_\infty(A) - \lambda$.*

Because in this paper the adversary is sometimes assumed to have some external information E about Alice and Bob's secrets, we need the following variant, defined in [DORS08, Definition 2], of the definition of strong extractors of [NZ96]:

Definition 1. *Let* $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^l$ *be a polynomial time probabilistic function that uses r bits of randomness. We say that* Ext *is an* average-case (n, m, l, ε)-strong extractor *if for all pairs of random variables (W, E) such that $w \in W$ is an n-bit string and $\widetilde{\mathbf{H}}_\infty(W \mid E) \geq m$, we have $\mathbf{SD}((\mathsf{Ext}(W; X), X, E), (U_l, X, E)) \leq \varepsilon$, where X is the uniform distribution over $\{0,1\}^r$.*

Any strong extractor can be made average-case with a slight increase in input entropy [DORS08, Section 2.5]. We should note that some strong extractors, such as universal hash functions [CW79, HILL99] discussed next, generalize without any loss to average-case.

The Leftover Hash Lemma We first recall the notion of universal hashing [CW79]:

Definition 2. *A family of efficient functions $\mathcal{H} = \{h_i : \{0,1\}^n \to \{0,1\}^\ell\}_{i \in I}$ is* universal *if for all distinct x, x' we have $\Pr_{i \leftarrow I}[h_i(x) = h_i(x')] \leq 2^{-l}$.*

$\mathcal{H}$ is pairwise independent *if for all distinct x, x' and all y, y' it holds that $\Pr_{i \in I}[h_i(x) = y \wedge h_i(x') = y'] \leq 2^{-2\ell}$.* ♢

Lemma 3 (Leftover Hash Lemma, average-case version [DORS08]). *For $\ell, m, \varepsilon > 0$, $\mathcal{H}$ is a strong (m, ε) average-case extractor (where the index of the hash function is the seed to the extractor) if $\mathcal{H}$ is universal and $\ell \leq m + 2 - 2\log\frac{1}{\varepsilon}$.*

This Lemma easily generalizes to the case when $\mathcal{H}$ is allowed to depend on the extra information E about the input X. In other words, every function in $\mathcal{H}$ takes an additional input e, and the family $\mathcal{H}$ is universal for every fixed value of e.

Secure Sketches and Fuzzy Extractors. We start by reviewing the definitions of secure sketches and fuzzy extractors from [DORS08]. Let $\mathcal{M}$ be a metric space with distance function dis (we will generally denote by n the length of each element in $\mathcal{M}$). Informally, a secure sketch enables recovery of a string $w \in \mathcal{M}$ from any "close" string $w' \in \mathcal{M}$ without leaking too much information about w.

Definition 3. *An* $(m, \tilde{m}, t)$-secure sketch *is a pair of efficient randomized procedures (*$\mathsf{SS}, \mathsf{SRec}$*) s.t.:*

1. *The sketching procedure* SS *on input* $w \in \mathcal{M}$ *returns a bit string* $s \in \{0,1\}^*$. *The recovery procedure* SRec *takes an element* $w' \in \mathcal{M}$ *and* $s \in \{0,1\}^*$.
2. Correctness: *If* $\mathsf{dis}(w, w') \le t$ *then* $\mathsf{SRec}(w', \mathsf{SS}(w)) = w$.
3. Security: *For any distribution* W *over* $\mathcal{M}$ *with min-entropy* m, *the (average) min-entropy of* W *conditioned on* s *does not decrease very much. Specifically, if* $\mathbf{H}_\infty(W) \ge m$ *then* $\widetilde{\mathbf{H}}_\infty(W \mid \mathsf{SS}(W)) \ge \tilde{m}$.

The quantity $m - \tilde{m}$ *is called the* entropy loss *of the secure sketch.* ♢

In this paper, we will construct a robust fuzzy extractor for the binary Hamming metric using secure sketches for the same metric. We will briefly review the syndrome construction from [DORS08, Construction 3] that we use (see also references therein for its previous incarnations). Consider an efficiently decodable $[n, n-k, 2t+1]$ linear error-correcting code C. The sketch $s = \mathsf{SS}(w)$ consists of the k-bit syndrome w with respect to C. We will use the fact that s is a (deterministic) linear function of w and that the entropy loss is at most $|s| = k$ bits in the construction of our robust fuzzy extractor for the Hamming metric.

We note that, as was shown in [DKRS06], the secure sketch construction for the set difference metric of [DORS08] can be used to extend the robust fuzzy extractor construction in the Hamming metric to the set difference metric.

While a secure sketch enables recovery of a string w from a close string w', a fuzzy extractor extracts a close-to-uniform string R and allows the precise reconstruction of R from any string w' close to w.

Definition 4. *An* $(m, \ell, t, \varepsilon)$-fuzzy extractor *is a pair of efficient randomized procedures (*$\mathsf{Gen}, \mathsf{Rep}$*) with the following properties:*

1. *The generation procedure* Gen, *on input* $w \in \mathcal{M}$, *outputs an extracted string* $R \in \{0,1\}^\ell$ *and a helper string* $P \in \{0,1\}^*$. *The reproduction procedure* Rep *takes an element* $w' \in \mathcal{M}$ *and a string* $P \in \{0,1\}^*$ *as inputs.*
2. Correctness: *If* $\mathsf{dis}(w, w') \le t$ *and* $(R, P) \leftarrow \mathsf{Gen}(w)$, *then* $\mathsf{Rep}(w', P) = R$.
3. Security: *For any distribution* W *over* $\mathcal{M}$ *with min-entropy* m, *the string* R *is close to uniform even conditioned on the value of* P. *Formally, if* $\mathbf{H}_\infty(W) \ge m$ *and* $(R, P) \leftarrow \mathsf{Gen}(W)$, *then we have* $\mathbf{SD}\left((R, P), U_\ell \times P\right) \le \varepsilon$. ♢

Note that fuzzy extractors allow the information P to be revealed to an adversary without compromising the security of the extracted random string R. However, they provide no guarantee when the adversary is active. Robust fuzzy extractors defined (and constructed) in [DKRS06] formalize the notion of security against active adversaries. We review the definition below.

If W, W' are two (correlated) random variables over a metric space $\mathcal{M}$, we say $\mathsf{dis}(W, W') \leq t$ if the distance between W and W' is at most t with probability one. We call (W, W') a (t, m)-*pair* if $\mathsf{dis}(W, W') \leq t$ and $\mathbf{H}_\infty(W) \geq m$.

Definition 5. *An $(m, \ell, t, \varepsilon)$-fuzzy extractor has* post-application (resp., pre-application) robustness δ *if for all (t, m)-pairs (W, W') and all adversaries $\mathcal{A}$, the probability that the following experiment outputs "success" is at most δ: sample (w, w') from (W, W'); let $(R, P) = \mathsf{Gen}(w)$; let $\tilde{P} = \mathcal{A}(R, P)$ (resp., $\tilde{P} = A(P)$); output "success" if $\tilde{P} \neq P$ and $\mathsf{Rep}(w', \tilde{P}) \neq \perp$.* ♢

We note that the above definitions can be easily extended to give *average-case* fuzzy extractors (where the adversary has some external information E correlated with W), and that our constructions satisfy those stronger definitions, as well.

3 The New Robust Extractor

In this section we present our new extractor with post-application robustness. We extend it to a robust *fuzzy* extractor in Section 5. Our approach is similar to that of [DKRS06]; a detailed comparison is given in Section 4.

Starting point: key agreement secure against a passive adversary. Recall that a strong extractor allows extraction of a string that appears uniform to an adversary even given the presence of the seed used for extraction. Therefore, a natural way of achieving key agreement in the errorless case is for Alice to pick a random seed i for a strong extractor and send it to Bob (in the clear). They could then use $R = \mathsf{Ext}(w; i)$ as the shared key. As long as the adversary is passive, the shared key looks uniform to her. However, such a protocol can be rendered completely insecure when executed in the presence of an active adversary because $\mathcal{A}$ could adversarially modify i to i' such that R' extracted by Bob has no entropy. To prevent such malicious modification of i we will require Alice to send an authentication of i (along with i) to Bob. In our construction, we authenticate i using w as the key and then extract from w using i as the seed. Details follow.

Construction. For the rest of the paper we will let $w \in \{0,1\}^n$. We will assume that n is even (if not, drop one bit of w, reducing its entropy by at most 1). To compute $\mathsf{Gen}(w)$, let a be the first half of w and b the second: $a = [w]_1^{n/2}, b = [w]_{n/2+1}^n$. View a,b as elements of $\mathbb{F}_{2^{n/2}}$. Let $v = n - m + \log \frac{1}{\delta}$, where δ is the desired robustness. Choose a random $i \in \mathbb{F}_{2^{n/2}}$. Compute $y = ia + b$. Let σ consist of the first v bits of y and the extracted key R consist of the rest of y: $\sigma = [y]_1^v$, $R = [y]_{v+1}^{n/2}$. Output $P = (i, \sigma)$.

$\mathsf{Gen}(w)$:
1. Let $a = [w]_1^{n/2}, b = [w]_{n/2+1}^{n}$
2. Select a random $i \leftarrow \mathbb{F}_{2^{n/2}}$
3. Set $\sigma = [ia + b]_1^v$, $R = [ia + b]_{v+1}^{n/2}$ and output $P = (i, \sigma)$

$\mathsf{Rep}(w, P' = (i', \sigma'))$:
1. Let $a = [w]_1^{n/2}, b = [w]_{n/2+1}^{n}$
2. If $\sigma' = [i'a + b]_1^v$ then compute $R' = [i'a + b]_{v+1}^{n/2}$ else output $\perp$

Theorem 1. *Let $\mathcal{M} = \{0,1\}^n$. Setting $v = n/2 - \ell$, the above construction is an $(m, \ell, 0, \varepsilon) -$ fuzzy extractor with robustness δ, for any $m, \ell, \varepsilon, \delta$ satisfying $\ell \le m - n/2 - \log \frac{1}{\delta}$ as long as $m \ge n/2 + 2 \log \frac{1}{\varepsilon}$.*

If ε is so low that the constraint $m \ge n/2 + 2 \log \frac{1}{\varepsilon}$ is not satisfied, then the construction can be modified as shown in Section 3.1.

Proof. Extraction. Our goal is to show that R is nearly uniform given P. To do so, we first show that the function $h_i(a, b) = (\sigma, R)$ is a universal hash family. Indeed, for $(a, b) \neq (a', b')$ consider

$$\begin{aligned}\Pr_i[h_i(a,b) = h_i(a',b')] &= \Pr_i[ia + b = ia' + b'] \\ &= \Pr_i[i(a - a') = (b - b')] \\ &\le 2^{-n/2}.\end{aligned}$$

To see the last inequality recall that $(a, b) \neq (a', b')$. Therefore, if $a = a'$, then $b \neq b'$ making the $\Pr_i[i(a - a') = (b - b')] = 0$. If $a \neq a'$, then there is a unique $i = (b - b')/(a - a')$ that satisfies the equality. Since i is chosen randomly from $\mathbb{F}_{2^{n/2}}$, the probability of the specific i occurring is $2^{-n/2}$.

Because $|(R, \sigma)| = n/2$, Lemma 3 gives us $\mathbf{SD}\left((R, P), U_{|R|} \times U_{|P|}\right) \le \varepsilon/2$ as long as $n/2 \le m + 2 - 2 \log \frac{2}{\varepsilon}$, or, equivalently, (R, P) is $2^{(n/2-m)/2-1}$-close to $U_{|R|} \times U_{|P|}$. Applying Lemma 1 to $A = R$, $B = P$, $C = U_{\frac{n}{2}-v}$, $D = U_{\frac{n}{2}} \times U_v$, we get that (R, P) is ε-close to $U_{(\frac{n}{2})-v} \times P$, for $\varepsilon = 2^{(n/2-m)/2}$. From here it follows that for extraction to be possible, $m \ge n/2 + 2 \log \frac{1}{\varepsilon}$.

Post-Application Robustness. In the post-application robustness security game, the adversary $\mathcal{A}$ on receiving $(P = (i, \sigma), R)$ (generated according to procedure Gen) outputs $P' = (i', \sigma')$, and is considered successful if $(P' \neq P) \wedge [i'a + b]_1^v = \sigma'$. In our analysis, we will assume that $i' \neq i$. We claim that this does not reduce $\mathcal{A}$'s success probability. Indeed, if $i' = i$ then, for $P' \neq P$ to hold, $\mathcal{A}$ would have to output $\sigma' \neq \sigma$. However, when $i' = i$, Rep would output $\perp$ unless $\sigma' = \sigma$.

In our analysis, we allow $\mathcal{A}$ to be deterministic. This is without loss of generality since we allow an unbounded adversary. We also allow $\mathcal{A}$ to arbitrarily fix i. This makes the result only stronger since we demonstrate robustness for a worst-case choice of i.

Since i is fixed and $\mathcal{A}$ is deterministic, (σ, R) determines the transcript $\mathsf{tr} = (i, \sigma, R, i', \sigma')$. For any particular tr, let $\mathsf{Succ}_{\mathsf{tr}}$ be the event that the transcript is tr and $\mathcal{A}$ wins, i.e., that $ia + b = \sigma || R \wedge [i'a + b]_1^v = \sigma'$. We denote by $\mathsf{Bad}_{\mathsf{tr}}$ the set of $w = a||b$ that make $\mathsf{Succ}_{\mathsf{tr}}$ true. For any tr, $\Pr_w[\mathsf{Succ}_{\mathsf{tr}}] \leq |\mathsf{Bad}_{\mathsf{tr}}| 2^{-m}$, because each w in $\mathsf{Bad}_{\mathsf{tr}}$ occurs with probability at most 2^{-m}. We now partition the set $\mathsf{Bad}_{\mathsf{tr}}$ into 2^ℓ disjoint sets, indexed by $R' \in \{0,1\}^\ell$:

$$\begin{aligned}\mathsf{Bad}_{\mathsf{tr}}^{R'} &\stackrel{\text{def}}{=} \{w \mid w \in \mathsf{Bad}_{\mathsf{tr}} \wedge [i'a + b]_{v+1}^{\ell} = R'\} \\ &= \{w \mid (ia + b = \sigma || R) \wedge (i'a + b = \sigma' || R')\}\end{aligned}$$

For a particular value of (tr, R'), $w = a||b$ is uniquely determined by the constraints that define the above set i.e; $|\mathsf{Bad}_{\mathsf{tr}}^{R'}| = 1$. Since $\mathsf{Bad}_{\mathsf{tr}} = \bigcup_{R' \in \{0,1\}^\ell} \mathsf{Bad}_{\mathsf{tr}}^{R'}$, we get $|\mathsf{Bad}_{\mathsf{tr}}| \leq 2^\ell = 2^{n/2-v}$. From here it follows that

$$\Pr[\mathsf{Succ}_{\mathsf{tr}}] \leq |\mathsf{Bad}_{\mathsf{tr}}| 2^{-m} \leq 2^{n/2-v-m}\,.$$

$\Pr[\mathsf{Succ}_{\mathsf{tr}}]$ measures the probability that the transcript is tr and $\mathcal{A}$ succeeds. To find out the probability that $\mathcal{A}$ succeeds, we need to simply add $\Pr[\mathsf{Succ}_{\mathsf{tr}}]$ over all possible tr. Since a transcript is completely determined by σ, R, the total number of possible transcripts is $2^{|\sigma|+|R|} = 2^{n/2}$ and, therefore, $\mathcal{A}$'s probability of success is at most 2^{n-v-m}.

To achieve δ-robustness, we need to set v to at least $n - m + \log \frac{1}{\delta}$. From here it follows that $\ell = \frac{n}{2} - v \leq \frac{1}{2}(2m - n - 2\log \frac{1}{\delta})$. □

3.1 Getting Closer to Uniform

If ε is so low that the constraint $m \geq n/2 + 2\log \frac{1}{\varepsilon}$ is not satisfied, then in our construction we can simply shorten R by $\beta = n/2 + 2\log \frac{1}{\varepsilon} - m$ bits, as follows: keep $v = n - m + \log \frac{1}{\delta}$ (regardless of ℓ), and let $R = [ia + b]_{v+1}^{\ell+v}$, for any $\ell \leq 2m - n - \log \frac{1}{\delta} - 2\log \frac{1}{\varepsilon}$. This keeps σ the same, but shortens R enough for the leftover hash lemma to work. The proof remains essentially the same, except that to prove robustness, we will give the remaining bits $[ia + b]_{\ell+v+1}^{n/2}$ for free to $\mathcal{A}$.

3.2 Improving the Construction of [DKRS06] When the Uniformity Constraint Dominates

The construction of Dodis et al. [DKRS06] parses w as two strings a and b of lengths $n - v$ and v, respectively. The values σ, R are computed as $\sigma = [ia]_1^v + b$ and $R = [ia]_{v+1}^n$; $P = (i, \sigma)$. In order to get R to be uniform given P, the value v is increased until the leftover hash lemma can be applied to (R, σ). However, we observe that this unnecessarily increases the length of σ (i.e., for every bit added to v, two bits are subtracted from R). Instead, we propose to improve this construction with essentially the same technique as we use for our construction in Section 3.1. The idea is to simply shorten R without increasing the length of σ. This improvement applies to both pre- and post-application robustness.

For post-application robustness, suppose the uniformity constraint dominates, i.e., $2\log\frac{1}{\varepsilon} > (2m-n+\log\frac{1}{\delta})/3$. Modify the construction of [DKRS06] by setting $v = (2n-m+\log\frac{1}{\delta})/3$ and $R = [ia]_{v+1}^{n-v-\beta}$, where $\beta = 2\log\frac{1}{\varepsilon} - (2m-n-\log\frac{1}{\delta})/3$. This will result in an extracted key of length $\ell = (4m-2n-\log\frac{1}{\delta})/3 - 2\log\frac{1}{\varepsilon}$. However, even with the improvement, the extracted key will be always shorter than the key extracted by our scheme, as explained in Section 4.2

In contrast, this improvement seems useful in the case of pre-application robustness. Again, suppose the uniformity constraint dominates, i.e., $2\log\frac{1}{\varepsilon} > \log\frac{1}{\delta}$. Modify the construction of [DKRS06] by setting $v = n - m + \log\frac{1}{\delta}$ and $R = [ia]_{v+1}^{n-v-\beta}$, where $\beta = 2\log\frac{1}{\varepsilon} - \log\frac{1}{\delta}$. This will result in an extracted key of length $\ell = 2m - n - 2\log\frac{1}{\varepsilon} - \log\frac{1}{\delta}$, which is $2\log\frac{1}{\varepsilon} - \log\frac{1}{\delta}$ longer than the key extracted without this modification.

4 Comparison with the Construction of [DKRS06]

4.1 When the Robustness Constraint Dominates

Recall that the construction of Dodis et al. [DKRS06] parses w as two strings a and b of lengths $n-v$ and v, respectively. The values σ, R are computed as $\sigma = [ia]_1^v + b$ and $R = [ia]_{v+1}^n$; $P = (i, \sigma)$. Notice that, like in our construction, increasing v improves robustness and decreases the number of extracted bits. For pre-application robustness, setting $v = n - m + \log\frac{1}{\delta}$ suffices, and thus the construction extracts nearly $(2m-n)$ bits. However, for post-application robustness, a much higher v is needed, giving only around $\frac{1}{3}(2m-n)$ extracted bits.

The post-application robustness game reveals more information to $\mathcal{A}$ about w than the pre-application robustness game. This additional information—namely, R—may make it easier for $\mathcal{A}$ to guess σ' for a well-chosen i'. The key to our improvement is in the pairwise independence of the function $ia+b$ that computes both σ and R: because of pairwise independence, the value (σ, R) of the function on input i tells $\mathcal{A}$ nothing about the value (σ', R') on another input i'. (This holds, of course, for uniformly chosen key (a, b); when (a, b) has entropy m, then $\mathcal{A}$ can find out $n-m$ bits of information about σ'.)

In contrast, in the construction of [DKRS06], only σ is computed using a pairwise independent hash function. This works well (in fact, better than our construction, because b can be shorter) for pre-application robustness, where $\mathcal{A}$ does not find out R. But it makes it possible for R to decrease $\mathcal{A}$'s uncertainty about σ' by as much as $\ell = |R|$, thus necessitating the length v of σ' (and hence σ) to be $v > \ell + (n-m)$ (the $(n-m)$ term is the amount of entropy already potentially "missing" from σ' because of the nonuniformity of w). See Section 4.3 for a detailed description of an adversarial strategy that utilizes R to obtain σ' in the [DKRS06] construction.

Another way to see the differences between the two constructions is through the proof. In the proof of post-application robustness, the transcript tr includes R, which makes for 2^ℓ times more transcripts than in the proof of pre-application robustness. However, the fact that this R imposes an additional constraint of w,

thus reducing the size of the set $\mathsf{Bad_{tr}}$, can compensate for this increase. It turns out that for the construction of [DKRS06], this additional constraint can be redundant if the adversary is clever about choosing i' and σ', and the size of $\mathsf{Bad_{tr}}$ doesn't decrease. Using a pairwise-independent function for computing R in our construction ensures that this additional constraint decreases the size of $\mathsf{Bad_{tr}}$ by 2^ℓ. Thus, our construction achieves the same results for pre- and post-application robustness.

4.2 When the Uniformity Constraint Dominates

It should be noted that there may be reasonable cases when the uniformity constraint ε on R is strong enough that the construction of [DKRS06] extracts even fewer bits, because it needs to take $v \geq n - m + 2\log\frac{1}{\varepsilon}$ to ensure near-uniformity of R given P. In that case, as long as $m \geq n/2 + 2\log\frac{1}{\varepsilon}$, our construction will extract the same amount of bits as before, thus giving it an even bigger advantage. And when $m < n/2 + 2\log\frac{1}{\varepsilon}$, our construction still extracts at least $3/2$ times more bits than the construction of [DKRS06], even with the improvement of Section 3.2 applied (this can be seen by algebraic manipulation of the relevant parameters for the post-application robustness case).

4.3 Why the Construction of [DKRS06] Cannot Extract More Bits

Recall that the robust fuzzy extractor of [DKRS06] operates as follows: parse w as two strings a, b of lengths $n - v, v$ respectively and compute $\sigma = [ia]_1^v + b$ and $R = [ia]_{v+1}^n$; $P = (i, \sigma)$.

For post-application robustness, the concern is that R can reveal information to the adversary about σ' for a cleverly chosen i'. Because the length of σ' is v and $\ell + (n - m)$ bits of information about σ' may be available (the ℓ term comes from $|R|$, and $(n - m)$ term comes from the part of w which has no entropy), this leads to the requirement that $v \geq \ell + n - m + \log\frac{1}{\delta}$ to make sure the adversary has to guess at least $\log\frac{1}{\delta}$ bits about σ'. Plugging in $\ell = n - 2v$, we obtain $\ell \leq \frac{2}{3}(m - n/2 - \log\frac{1}{\delta})$, which is the amount extracted by the construction.

Here we show an adversarial strategy that indeed utilizes R to obtain information about σ' to succeed with probability $\delta/2$. This demonstrates that the analysis in [DKRS06] is tight up to one bit. To do so we have to fix a particular (and somewhat unusual) representation of field elements. (Recall that any representation of field elements works for constructions here and in [DKRS06], as long as addition of field elements corresponds to the exclusive-or of bit strings.) Typically, one views $\mathbb{F}_{2^{n-v}}$ as $\mathbb{F}_2[x]/(p(x))$ for some irreducible polynomial p of degree $n - v$, and represents elements as $\mathbb{F}_2$-valued vectors in the basis $(x^{n-v-1}, x^{n-v-2}, ..., x^2, x, 1)$. We will do the same, but will reorder the basis elements so as to separate the even and the odd powers of x: $(x^{n-v-1}, x^{n-v-3}, \ldots, x, x^{n-v-2}, x^{n-v-4}, \ldots, 1)$ (assuming, for concreteness, that $n - v$ is even). The advantage of this representation for us is that the top half of bits of some value $z \in F_{2^{n-v}}$ is equal to the bottom half of the bits of z/x, as long as the last bit of z is 0.

Now suppose the distribution on w is such that the top $n-m$ bits of b are 0 (the rest of the bits of w are uniform). Then by receiving σ and R, the adversary gets to see the top $\ell+(n-m)$ bits of ia. Therefore, the adversary knows $\ell+(n-m)$ bits from the bottom half of ia/x as long as the last bit of ia is 0, which happens with probability 1/2. To use this knowledge, the adversary will simply ensure that the difference between σ' and σ is $[ia/x]_1^v$, by letting $i' = i + i/x$.

Thus, the adversarial strategy is as follows: let $i' = i + i/x$; let τ consist of the ℓ bits of R, the top $n-m$ bits of σ, and $\log\frac{1}{\delta} = v - \ell - (n-m)$ randomly guessed bits, and let $\sigma' = \sigma + \tau$. The adversary wins whenever $\tau = [ia/x]_1^v$, which happens with probability $2^{v-\ell-(n-m)}/2 = \delta/2$, because all but $\log\frac{1}{\delta}$ bits of τ are definitely correct as long as the last bit of ia is 0.

The above discussion gives us the following result.

Theorem 2. *There exists a basis for $GF(2^{n-v})$ such that for any integer m there exists a distribution W of min-entropy m for which the post-application robustness of the construction from [DKRS06, Theorem 3] can be violated with probability at least $\delta/2$, where v is set as required for robustness δ by the construction (i.e., $v = (n-\ell)/2$ for $\ell = (2m - n - 2\log\frac{1}{\delta})/3$).*

Note that our lower bound uses a specific representation of field elements, and hence does not rule out that for some particular representation of field elements, a lower value of v and, therefore, a higher value of ℓ is possible. However, a security proof for a lower value of v would have to then depend on the properties of that particular representation and would not cover the construction of [DKRS06] in general.

5 Tolerating Binary Hamming Errors

We now consider the scenario where Bob has a string w' that is close to Alice's input w (in the Hamming metric). In order for them to agree on a random string, Bob would first have recover w from w'. To this end, Alice could send the secure sketch $s = \mathsf{SS}(w)$ to Bob along with (i, σ). To prevent an undetected modification of s to s', she could send an authentication of s (using w as the key) as well. The nontriviality of making such an extension work arises from the fact that modifying s to s' also gives the adversary the power to influence Bob's verification key $w^* = \mathsf{SRec}(w', s')$. The adversary could perhaps exploit this circularity to succeed in an active attack (the definition of standard authentication schemes only guarantee security when the keys used for authentication and verification are the same).

We break this circularity by exploiting the algebraic properties of the Hamming metric space, and using authentication secure against algebraic manipulation [DKRS06, CDF+08]. The techniques that we use are essentially the same as used in [DKRS06], but adapted to our construction. We present the construction here and then discuss the exact properties that we use in the proof of security.

Construction. Let $\mathcal{M}$ be the Hamming metric space on $\{0,1\}^n$. Let W be a distribution of min-entropy m over $\mathcal{M}$. Let $s = \mathsf{SS}(w)$ be a deterministic, linear secure sketch; let $|s| = k$, $n' = n - k$. Assume that SS is a surjective linear function (which is the case for the syndrome construction for the Hamming metric mentioned in Section 2). Therefore, there exists a $k \times n$ matrix S of rank k such that $\mathsf{SS}(w) = \mathsf{S}w$. Let $\mathsf{S}^\perp$ be an $n' \times n$ matrix such that $n \times n$ matrix $\left(\frac{\mathsf{S}}{\mathsf{S}^\perp}\right)$ has full rank. We let $\mathsf{SS}^\perp(w) = \mathsf{S}^\perp(w)$.

To compute $\mathsf{Gen}(w)$, let $s = \mathsf{SS}(w)$, $c = \mathsf{SS}^\perp(w)$; $|c| = n'$. We assume that n' is even (if not, drop one bit of c, reducing its entropy by at most 1). Let a be the first half of c and b the second. View a, b as elements of $\mathbb{F}_{2^{n'/2}}$. Let $L = 2\lceil \frac{k}{n'} \rceil$ (it will important for security that L is even). Pad s with 0s to length $Ln'/2$, and then split it into L bit strings $s_{L-1}, \ldots, s_0$ of length $n'/2$ bits each, viewing each bit string as an element of $\mathbb{F}_{2^{n'/2}}$. Select $i \leftarrow \mathbb{F}_{2^{n'/2}}$. Define $f_{s,i}(x) = x^{L+3} + x^2(s_{L-1}x^{L-1} + s_{L-2}x^{L-2} + \cdots + s_0) + ix$. Set $\sigma = [f_{s,i}(a) + b]_1^v$, and output $P = (s, i, \sigma)$ and $R = [f_{s,i}(a) + b]_{v+1}^{n'/2}$.

$\mathsf{Gen}(w)$:

1. Set $s = \mathsf{SS}(w)$, $c = \mathsf{SS}^\perp(w)$, $k = |s|$, $n' = |c|$.
   - Let $a = [c]_1^{n'/2}$, $b = [c]_{n'/2+1}^{n'}$
   - Let $L = 2\lceil \frac{k}{n'} \rceil$. Pad s with 0s to length $Ln'/2$.
   - Parse the padded s as $s_{L-1}||s_{L-2}||\ldots||s_0$ for $s_i \in \mathbb{F}_{2^{n'/2}}$.
2. Select $i \leftarrow \mathbb{F}_{2^{n'/2}}$.
3. Set $\sigma = [f_{s,i}(a) + b]_1^v$, and output $R = [f_{s,i}(a) + b]_{v+1}^{n'/2}$ and $P = (s, i, \sigma)$.

$\mathsf{Rep}(w', P' = (s', i', \sigma'))$:

1. Compute $w^* = \mathsf{SRec}(w', s')$
   - Verify that $\mathsf{dis}(w^*, w') \le t$ and $\mathsf{SS}(w^*) = s'$. If not, output $\perp$.
2. Let $c' = \mathsf{SS}^\perp(w^*)$. Parse c' as $a'||b'$.
3. Compute $\sigma^* = [f_{s',i'}(a') + b']_1^v$.
   - Verify that $\sigma^* = \sigma'$. If so, output $R = [f_{s',i'}(a') + b']_{v+1}^{n'/2}$, else output $\perp$.

In the theorem statement below, let B denote the volume of a Hamming ball or radius t in $\{0,1\}^n$ ($\log B \le nH_2(t/n)$ [MS77, Chapter 10, §11, Lemma 8] and $\log B \le t \log(n+1)$ [DKRS06]).

Theorem 3. *Assume* SS *is a deterministic linear* $(m, m-k, t)-$*secure sketch of output length* k *for the Hamming metric on* $\{0,1\}^n$. *Setting* $v = (n-k)/2 - l$, *the above construction is an* (m, l, t, ε) *fuzzy extractor with robustness* δ *for any* m, l, t, ε *satisfying* $l \le m - n/2 - k - \log B - \log\left(2\left\lceil \frac{k}{n-k} \right\rceil + 2\right) - \log \frac{1}{\delta}$ *as long as* $m \ge \frac{1}{2}(n+k) + 2\log\frac{1}{\varepsilon}$.

Again, if $m < \frac{1}{2}(n+k) + 2\log\frac{1}{\varepsilon}$, the construction can be modified, as shown in Section 5.1.

Proof. Extraction. Our goal is to show that R is nearly uniform given $P = (i, s, \sigma)$. To do so, we first note that for every s, the function $h_i(c) = (\sigma, R)$ is a

universal hash family. Indeed for $c \neq c'$ there is a unique i such that $h_i(c) = h_i(c')$ (since $i(a - a')$ is fixed, like in the errorless case). We also note that $\widetilde{\mathbf{H}}_\infty(c \mid \mathsf{SS}(W)) \geq \widetilde{\mathbf{H}}_\infty(c, \mathsf{SS}(W)) - k = \mathbf{H}_\infty(W) - k = m - k$ by Lemma 2. Because $|(R, \sigma)| = n'/2$, Lemma 3 (or, more precisely, its generalization mentioned in the paragraph following the lemma, needed here because h_i depends on s) gives us

$$\mathbf{SD}\left((R, P), U_{|R|} \times \mathsf{SS}(W) \times U_{n'/2} \times U_v\right) \leq \varepsilon/2$$

for $n'/2 \leq m - k + 2 - 2\log(2/\varepsilon)$. This is equivalent to saying that (R, P) is $2^{(n'/2-m+k)\frac{1}{2}-1}$-close to $U_{|R|} \times \mathsf{SS}(W) \times U_{n'/2} \times U_v$.

Applying Lemma 1 to $A = R$, $B = P$, $C = U_{n'/2-v}$, $D = \mathsf{SS}(w) \times U_{n'/2} \times U_v$, we get that (R, P) is ε-close to $U_{\frac{n'}{2}-v} \times P$, for $\varepsilon = 2^{(\frac{n'}{2}-m+k)/2}$.

From here it follows that for extraction to be possible, $m \geq \frac{1}{2}(n+k)+2\log\frac{1}{\varepsilon}$.

Post-Application Robustness. In the post-application robustness security game, the adversary $\mathcal{A}$ on receiving $(P = (s, i, \sigma), R)$ (generated according to procedure Gen) outputs $P' = (s', i', \sigma')$, and is considered successful if $(P' \neq P) \wedge \mathsf{Rep}(w', s') \neq \perp$. In our analysis, we will assume that $(i', s') \neq (i, s)$. We claim that this does not reduce $\mathcal{A}$'s success probability. Indeed, if $(i', s') = (i, s)$ then, c' computed within Rep will equal c. So, for $P' \neq P$ to hold, $\mathcal{A}$ would have to output $\sigma' \neq \sigma$. However, when $(i', c', s') = (i, c, s)$, Rep would compute $\sigma^* = \sigma$, and therefore would output $\perp$ unless $\sigma' = \sigma$.

In our analysis, we allow $\mathcal{A}$ to be deterministic. This is without loss of generality since we allow an unbounded adversary. We also allow $\mathcal{A}$ to arbitrarily fix i. This makes the result only stronger since we demonstrate robustness for a worst-case choice of i.

Since i is fixed and $\mathcal{A}$ is deterministic, the $\mathsf{tr} = (i, s, \sigma, R, i', s', \sigma')$ is determined completely by (s, σ, R). Recall that the prime challenge in constructing a robust fuzzy extractor was that $\mathcal{A}$ could somehow relate the key used by Rep to verify σ' to the authentication key that was used by Gen to come up with σ. As was done in [DKRS06], we will argue security of our construction by showing that the MAC scheme implicitly used in our construction remains unforgeable even when $\mathcal{A}$ could force the verification key to be at an offset (of her choice) from the authentication key. We will formalize such an argument by assuming that $\mathcal{A}$ learns $\Delta = w' - w$. Recall that $w^* = \mathsf{SRec}(w', s')$ and $c' = a'||b' = \mathsf{SS}^{\perp}(w^*)$. The following claim that was proven in [DKRS06] states that given (Δ, s), $\mathcal{A}$ can compute the offsets $\Delta_a = a' - a$, $\Delta_b = b' - b$ induced by her choice of s'.

Claim. Given $\Delta = w' - w$, and the sketches $s, s', \mathcal{A}$ can compute $\Delta_a = a' - a$ and $\Delta_b = b' - b$, or determine that Rep will reject before computing a', b'.

In other words, she can compute the offset between the authentication key that Gen used to come up with σ and the verification key that Rep will use to verify σ'. We will now argue that as long as W has sufficient min-entropy, even knowing the offset does not help $\mathcal{A}$ succeed in an active attack. Recall that since i is arbitrarily fixed by $\mathcal{A}$, $\mathcal{A}$'s success depends on w, w', or, alternatively, on w, Δ. Fix some Δ. For any particular tr, let $\mathsf{Succ}_{\mathsf{tr},\Delta}$ be the event that the transcript

is tr and $\mathcal{A}$ wins, i.e., that $f_{s,i}(a) + b = \sigma||R \wedge [f_{s',i'}(a') + b']_1^v = \sigma' \wedge \mathsf{SS}(w) = s$, conditioned on the fact that $w' - w$ is Δ. We denote by $\mathsf{Bad}_{\mathsf{tr},\Delta}$ the set of w that make $\mathsf{Succ}_{\mathsf{tr},\Delta}$ true. We now partition the set $\mathsf{Bad}_{\mathsf{tr},\Delta}$ into 2^ℓ disjoint sets, indexed by $R' \in \{0,1\}^\ell$:

$$\begin{aligned}\mathsf{Bad}_{\mathsf{tr},\Delta}^{R'} &\stackrel{\text{def}}{=} \{w \,|\, w \in \mathsf{Bad}_{\mathsf{tr},\Delta} \wedge [f_{s',i'}(a') + b']_{v+1}^{\ell} = R'\} \\ &= \{w \,|\, (f_{s,i}(a) + b = \sigma||R) \wedge (f_{s',i'}(a') + b' = \sigma'||R') \wedge \mathsf{SS}(w) = s\}.\end{aligned}$$

By Claim 1, fixing (tr, Δ), also fixes Δ_a, Δ_b. It follows that every $w \in \mathsf{Bad}_{\mathsf{tr},\Delta}^{R'}$ needs to satisfy

$$f_{s,i}(a) - f_{s',i'}(a + \Delta_a) = (\Delta_b + \sigma - \sigma')||(R - R') \wedge \mathsf{SS}(w) = s.$$

For a given tr, Δ, R', the right hand side of the first equation takes a fixed value. Let us now focus on the polynomial $f_{s,i}(a) - f_{s',i'}(a + \Delta_a)$. We will consider two cases:

- $\Delta_a = 0$: In this case, $f_{s,i}(x) - f_{s',i'}(x)$ is a polynomial in which a coefficient of degree 2 or higher is nonzero if $s \neq s'$ and a coefficient of degree 1 or higher is nonzero if $i \neq i'$.
- $\Delta_a \neq 0$: Observe that the leading term of the polynomial is $((L+3) \bmod 2)\Delta_a x^{L+2}$. Since we forced L to be even, the coefficient of the leading term is nonzero, making $f_{s,i}(x) - f_{s',i'}(x + \Delta_a)$ a polynomial of degree $L + 2$.

Therefore, in either case, the $f_{s,i}(x) - f_{s',i'}(x + \Delta_a)$ is a nonconstant polynomial of degree at most $L + 2$. A nonconstant polynomial of degree d can take on a fixed value at most d times. It, therefore, follows that there are at most $L + 2$ values of a such that $f_{s,i}(a) - f_{s',i'}(a + \Delta_a) = (\Delta_b + \sigma - \sigma')||(R - R')$. Each such a uniquely determines $b = (\sigma||R) - f_{s,i}(a)$. And w is uniquely determined by $c = a||b = \mathsf{SS}^{\perp}(w)$ and $s = \mathsf{SS}(w)$. Therefore, there are at most $L + 2$ values of w in the set $\mathsf{Bad}_{\mathsf{tr},\Delta}^{R'}$ i.e, $|\mathsf{Bad}_{\mathsf{tr},\Delta}^{R'}| \leq L + 2$. Since $\mathsf{Bad}_{\mathsf{tr},\Delta} = \bigcup_{R' \in \{0,1\}^\ell} \mathsf{Bad}_{\mathsf{tr},\Delta}^{R'}$, we get $|\mathsf{Bad}_{\mathsf{tr},\Delta}| \leq (L+2)2^\ell = (L+2)2^{n'/2-v}$. Thus, $\Pr_w[\mathsf{Succ}_{\mathsf{tr},\Delta}] \leq |\mathsf{Bad}_{\mathsf{tr}}|2^{-\mathbf{H}_\infty(w|\Delta)} \leq (L+2)2^{n'/2-v-\mathbf{H}_\infty(w|\Delta)}$.

To find out the probability $\Pr_w[\mathsf{Succ}_\Delta]$ that $\mathcal{A}$ succeeds conditioned on a particular Δ, we need to add up $\Pr_w[\mathsf{Succ}_{\mathsf{tr},\Delta}]$ over all possible transcripts. Recalling that each transcript is determined by σ, R and s and hence there are $2^{n'/2+k}$ of them, and that $n' + k = n$, we get $\Pr_w[\mathsf{Succ}_\Delta] \leq (L+2)2^{n-v-\mathbf{H}_\infty(w|\Delta)}$.

Finally, the probability of adversarial success it at most

$$\mathop{\mathbf{E}}_{\Delta} \Pr_w[\mathsf{Succ}_\Delta] \leq (L+2)2^{n-v-\widetilde{\mathbf{H}}_\infty(w|\Delta)}.$$

In particular, if the errors Δ are independent of w, then $\widetilde{\mathbf{H}}_\infty(w|\Delta) = \mathbf{H}_\infty(w) = m$, and the probability of adversarial success is at most $(L+2)2^{n-v-m}$. In the worst case, however, the entropy of w may decrease at most by the number of bits needed to represent Δ. Let B be the volume of the hamming ball of radius t

in $\{0,1\}^n$. Then, Δ can be represented in $\log B$ bits and $\widetilde{\mathbf{H}}_\infty(w|\Delta) \geq m - \log B$, by Lemma 2. From here it follows that

$$\Pr[\mathcal{A}'s\ success] \leq B(L+2)2^{n-v-m}$$

To achieve $\delta-$robustness, we want $B(L+2)2^{n-v-m} \leq \delta$ i.e., $v \geq n-m+\log B+\log(L+2)+\log\frac{1}{\delta}$. Setting $v = n-m+\log B+\log(L+2)+\log\frac{1}{\delta}$, and using $L = 2\lceil\frac{k}{n-k}\rceil$ it follows that

$$\ell \leq m - n/2 - k - \log B - \log\left(2\left\lceil\frac{k}{n-k}\right\rceil + 2\right) - \log\tfrac{1}{\delta}\,. \qquad \square$$

5.1 Getting Closer to Uniform

If ε is so low that $m \geq \frac{1}{2}(n+k) + 2\log\frac{1}{\varepsilon}$ does not hold, we can modify our construction just as we did in section 3.1, by shortening R by $\beta = \frac{1}{2}(n+k) + 2\log\frac{1}{\varepsilon} - m$. That is, keep $v = n-m+\log B+\log(L+2)+\log\frac{1}{\delta}$ fixed and let $R = [f_{s,i}(a)+b]_{v+1}^{\ell+v}$, where $\ell \leq n/2 - v - \beta$.

References

[BBCM95] Bennett, C.H., Brassard, G., Crépeau, C., Maurer, U.M.: Generalized privacy amplification. IEEE Transactions on Information Theory 41(6), 1915–1923 (1995)

[BBR88] Bennett, C., Brassard, G., Robert, J.: Privacy amplification by public discussion. SIAM Journal on Computing 17(2), 210–229 (1988)

[BDK+05] Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 147–163. Springer, Heidelberg (2005)

[CDF+08] Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 471–488. Springer, Heidelberg (2008)

[CW79] Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences 18, 143–154 (1979)

[DKRS06] Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust fuzzy extractors and authenticated key agreement from close secrets. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 20–24. Springer, Heidelberg (2006)

[DKRS08] Dodis, Y., Katz, J., Reyzin, L., Smith, A.: Robust fuzzy extractors and authenticated key agreement from close secrets. Manuscript (2008)

[DORS08] Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing 38(1), 97–139 (2008)

[DS02] Dodis, Y., Spencer, J.: On the (non-)universality of the one-time pad. In: 43rd Annual Symposium on Foundations of Computer Science, pp. 376–385. IEEE, Los Alamitos (2002)

[HILL99] Hrastad, J., Impagliazzo, R., Levin, L.A., Luby, M.: Construction of pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)

[Mau93] Maurer, U.: Protocols for secret key agreement by public discussion based on common information. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 461–470. Springer, Heidelberg (1994)

[Mau97] Maurer, U.: Information-theoretically secure secret-key agreement by NOT authenticated public discussion. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 209–225. Springer, Heidelberg (1997)

[MS77] MacWilliams, F.J., Sloane, N.J.A.: The Theory of Error-Correcting Codes. North-Holland Elsevier Science (1977)

[MW97] Maurer, U., Wolf, S.: Privacy amplification secure against active adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 307–321. Springer, Heidelberg (1997)

[MW03] Maurer, U., Wolf, S.: Secret-key agreement over unauthenticated public channels — Part III: Privacy amplification. IEEE Trans. Info. Theory 49(4), 839–851 (2003)

[NZ96] Nisan, N., Zuckerman, D.: Randomness is linear in space. Journal of Computer and System Sciences 52(1), 43–53 (1996)

[PRTG02] Pappu, R., Recht, B., Taylor, J., Gershenfeld, N.: Physical one-way functions. Science 297, 2026–2030 (2002)

[RW03] Renner, R., Wolf, S.: Unconditional authenticity and privacy from an arbitrarily weak secret. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 78–95. Springer, Heidelberg (2003)

[RW04] Renner, R., Wolf, S.: The exact price for unconditionally secure asymmetric cryptography. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 109–125. Springer, Heidelberg (2004)

[WC81] Wegman, M.N., Carter, J.L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22, 265–279 (1981)

[Wic08] Wichs, D.: Private Communication (2008)

[Wol98] Wolf, S.: Strong security against active attacks in information-theoretic secret-key agreement. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 405–419. Springer, Heidelberg (1998)

[Wyn75] Wyner, A.D.: The wire-tap channe. Bell System Technical Journal 54(8), 1355–1387 (1975)

On Linear Secret Sharing for Connectivity in Directed Graphs

Amos Beimel[1] and Anat Paskin[2]

[1] Dept. of computer science, Ben-Gurion University, Beer Sheva, Israel
[2] Dept. of computer science, Technion, Haifa, Israel

Abstract. In this work we study linear secret sharing schemes for s-t connectivity in directed graphs. In such schemes the parties are edges of a complete directed graph, and a set of parties (i.e., edges) can reconstruct the secret if it contains a path from node s to node t. We prove that in every linear secret sharing scheme realizing the st-con function on a directed graph with n edges the total size of the shares is $\Omega(n^{1.5})$. This should be contrasted with s-t connectivity in undirected graphs, where there is a scheme with total share size n. Our result is actually a lower bound on the size monotone span programs for st-con, where a monotone span program is a linear-algebraic model of computation equivalent to linear secret sharing schemes. Our results imply the best known separation between the power of monotone and non-monotone span programs. Finally, our results imply the same lower bounds for matching.

1 Introduction

Secret sharing schemes, introduced by [11,35,26], are a method in which a dealer holding a secret can distribute shares to parties in a network such that only predefined authorized sets of parties can reconstruct the secret from their shares. These schemes, whose original motivation was secure storage, have found numerous applications as a building box in complex cryptographic schemes, e.g., Byzantine agreement [32], secure multiparty computations [8,16,17], threshold cryptography [20], access control [30], and attribute based encryption [25]. In most applications it is important that the scheme is linear, that is, the shares are a linear combination of the secret and some random elements. Linear secret sharing schemes are equivalent to monotone span programs, a computational model introduced by Karchmer and Wigderson [28].

In this work we study linear secret sharing schemes for a natural function: the parties are edges of a complete *directed* graph, and a set of parties (i.e., edges) is authorized if it contains a path from node s to node t. We prove that in every linear secret sharing scheme realizing the st-con function on a directed graph with n edges the total size of the shares is $\Omega(n^{1.5})$. Studying linear secret sharing for this function has both a cryptographic motivation and a computational complexity motivation. We first discuss the cryptographic motivation. Benaloh

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 172–184, 2008.

and Rudich [10] (see also [4,28]) showed that there exists a simple and very efficient linear secret sharing scheme for the analogous function where the graph is *undirected*. This scheme was used in [30] to design a protocol for reliable access control. The obvious open problem is if this scheme can be generalized to deal with directed graphs. The computational complexity motivation is separating the power of monotone and non-monotone span programs. Our results imply that over infinite fields and large finite fields non-monotone span programs are more efficient than monotone span programs by a multiplicative factor of $\Omega(n^{0.5})$. This is the best separation known to-date.

1.1 Previous Results

In this section we will give a short background on secret sharing schemes, linear secret sharing schemes, monotone span programs, and the equivalence of the latter two notions. Finally, we will discuss some known results on the s-t connectivity function.

Secret-sharing schemes were first introduced by Blakley [11] and Shamir [35] for the threshold case, that is, for the case where the subsets that can reconstruct the secret are all the sets whose cardinality is at least a certain threshold. Secret-sharing schemes for general access structures were introduced by Ito, Saito, and Nishizeki [26]. More efficient schemes were presented in, e.g., [9,36,14,28,37,22]. Even with the more efficient schemes, the size of the shares for general access structures with n parties is $\ell 2^{O(n)}$, where the secret is an ℓ-bit string. Lower bounds for secret sharing schemes were proved in [29,9,15,13,21,18,19,12,31]. The best lower bound was proved by Csirmaz [18], proving that, for every n, there is an access structure with n parties such that sharing an ℓ-bit secrets requires shares of length $\Omega(\ell n/\log n)$. Still there is an exponential gap between the lower-bounds and the upper-bounds.

Span programs and monotone span programs, introduced by Karchmer and Wigderson [28], are linear-algebraic models of computation. More specifically, a monotone span program is presented as a matrix over some field, with rows labeled by variables. The span program accepts an input if the rows whose variables are satisfied by the input span a fixed nonzero vector. The size of a span program is its number of rows. A detailed definition is given in Section 2. Lower bounds for monotone span programs have been studied in several papers. Beimel, Gál, and Paterson [6] provided a technique for proving lower bounds for monotone span programs and proved a lower bound of $O(n^{2.5})$ for a function with n variables. Babai, Gál, and Wigderson [2], using the technique of [6], proved the first super-polynomial lower bound – they prove an $n^{\Omega(\log n/\log\log n)}$ lower bound for the size of monotone span programs for the clique problem. Gál [23] gave a characterization of span program size and improved the lower bound for the clique function to $n^{\Omega(\log n)}$. Proving exponential lower bounds for an explicit function is an open problem (it is known that such lower bound holds for most functions [34]). Gál and Pudlák [24] have shown limitations of current techniques for proving lower bounds for monotone span programs. Beimel and Weinreb [7] showed a separating of the power of monotone span programs over

different fields, for example, they showed that there are functions that have small monotone span program over the field GF(2), however, they require super polynomial monotone span programs over fields whose characteristic is not 2.

In most applications of secret sharing schemes it is important that the scheme is linear, that is, the shares are a linear combination of the secret and some random elements. Linearity implies that if we sum shares distributed for two secrets, then we get shares of the sum of the secrets. This property is useful, for example, when designing secure multi-party protocols [8,16,17]. Karchmer and Wigderson [28] showed that monotone span programs imply linear secret sharing schemes (this result was implicitly proved also by Brickell [14]). More precisely, if there is a monotone span of size s computing a function f over a field $\mathbb{F}$ then there is a secret sharing scheme realizing f such that the domain of secrets is $\mathbb{F}$ and the total number of bits of the shares is $s \log |\mathbb{F}|$. In fact, monotone span programs and linear secret sharing schemes are equivalent [3]. Thus, proving lower bounds for monotone span programs implies the same lower bounds for linear secret sharing schemes.

In this work we prove lower bounds for the st-con function. This function is widely studied in complexity both for directed and undirected graphs. For example, st-con in directed graphs is NL-complete, while Reingold [33] has proved that st-con in undirected graphs is in deterministic log-space. Another example where undirected st-con is easier than directed st-con was given by Ajtai and Fagin [1]; they showed that while undirected st-con is definable in monadic second order logic, the directed case is not. We continue this line of research by proving that for monotone span programs undirected st-con is easier than directed st-con, although the gap we can prove is much smaller.

The circuit complexity of st-con has been studied as well. The directed (and undirected) st-con function has a polynomial-size monotone circuit of depth $O(\log n)$; this circuit has unbounded fan-in. This implies a monotone formula for st-con of size $n^{O(\log n)}$ and, using the construction of Benaloh and Leichter [9], there is a secret sharing scheme realizing the st-con function in which the size of the shares is $n^{O(\log n)}$. Karchmer and Wigderson [27] have proved that for monotone formulae this is optimal – every monotone formula computing undirected (and, hence, directed) st-con function has size $n^{\Omega(\log n)}$.

1.2 Our Results

In this work we prove that a monotone span program computing the st-con function on a *directed* graph with n edges has size $\Omega(n^{1.5})$. We supply two proofs of this lower bound. The first proof uses the characterization of span program size given by Gál [23]; this proof only holds for finite fields. The second proof uses the condition of Beimel, Gál, and Paterson [6]; this proof holds for every field. As monotone span program are equivalent to linear secret sharing schemes, our result implies that in every linear secret sharing scheme realizing the st-con function in directed graphs, the total size of the shares is $\Omega(n^{1.5})$.

Our lower bound has a few additional implications. First, it shows that, for monotone span programs and linear secret sharing, undirected st-con is easier

than directed st-con. This is true since there is a monotone span program realizing undirected st-con whose size is n [10,28] (see Example 1 below).

Furthermore, our lower bound supplies the best known separation between the power of monotone and non-monotone span programs. Beimel and Gál [5] proved that over infinite fields and large finite fields the directed st-con function on graphs with n edges has a *non-monotone* span program of size $O(n)$. Thus, our result shows a separation of multiplicative factor of $\Omega(n^{0.5})$ between monotone and non monotone span programs for directed st-con. Separations between monotone and non-monotone models of computation is an important question in complexity, e.g., the exponential separation between the power of monotone and non-monotone circuits [38]. Separations between the power of monotone and non-monotone span programs is interesting since monotone span programs can be exponentially more powerful than monotone circuits [2].

Finally, our result implies the same lower bound for matching and bipartite matching. This follows from the projection reduction from directed st-con to bipartite matching. Babai, Gál, and Wigderson [2] constructed a non-monotone span program, over large enough fields, for matching whose size is n (where n is the number of edges in the graph). Thus, the same separation between monotone and non-monotone span programs holds for matching.

1.3 Organization

In Section 2 we define monotone span programs. In Section 3 we give our first proof of the lower bound and in Section 4 we give our second proof of the lower bound.

2 Preliminaries

2.1 Monotone Span Programs

We start with the definition of monotone span programs. As discussed above, monotone span programs are equivalent to linear secret sharing schemes; we use monotone span programs to prove lower bounds on linear secret sharing schemes.

Definition 1 (Monotone Span Program [28]). *A* monotone span program *over a field $\mathbb{F}$ is a triplet $\widehat{M} = \langle M, \rho, \boldsymbol{v} \rangle$, where M is a matrix over $\mathbb{F}$, $\boldsymbol{v}$ is a nonzero row vector called the target vector (it has the same number of coordinates as the number of columns in M), and ρ is a labeling of the rows of M by variables from $\{x_1, \ldots, x_n\}$ (every row is labeled by one variables, and the same variable can label many rows).*

A monotone span program accepts or rejects an input by the following criterion. For every input $u \in \{0,1\}^n$ define the sub-matrix M_u of M consisting of those rows whose labels are satisfied by the assignment u. The monotone span program $\widehat{M}$ accepts *u if and only if $\boldsymbol{v} \in \text{span}(M_u)$, i.e., some linear combination of the rows of M_u gives the target vector $\boldsymbol{v}$. A monotone span program* computes

a Boolean function f if it accepts exactly those inputs u where $f(u) = 1$. The size *of $\widehat{M}$ is the number of rows in M.*[1]

Monotone span programs compute only monotone functions, and every monotone Boolean function can be computed by a monotone span program. The size of the smallest monotone span program over $\mathbb{F}$ that computes f is denoted by $\mathrm{mSP}_{\mathbb{F}}(f)$.

Example 1. Consider the undirected-st-con function, whose input is an undirected graph with two designated nodes s and t and its output is 1 iff the graph contains a path from s to t. Formally, we consider the following function: The input is an undirected graph with $m+2$ nodes; the variables of the function are the $n = \binom{m+2}{2}$ possible edges. Karchmer and Wigderson [28] construct a monotone span program of size n for this function, that is, each edge labels exactly one row in the program (a linear secret sharing scheme equivalent to this program was previously shown in [10]).

We next describe this span program. Assume the nodes of the input graph are $z_0, \ldots, z_{m+1}$, where $z_0 = s$ and $z_{m+1} = t$. The program has $m+2$ columns and n rows. For every edge (z_i, z_j), where $i < j$, there is a row in the program; in this row all entries in the row are zero, except for the ith entry which is 1 and the jth entry which is -1. The target vector is the same as the row labeled by (s,t), that is, $(1, 0, \ldots, 0, -1)$. It can be proven that over every field $\mathbb{F}$, this program computes the undirected-st-con function.

2.2 The st-con Function

In the rest of the paper we will refer to the st-con function in directed graphs as st-con. Formally, we consider the following function: The input is a directed graph with $m+2$ nodes. The graph contains two designated nodes s, t. The variables are the $n = m(m+1)$ possible edges in the graph. The function outputs 1 iff there is a directed path from node s to node t. Our main results are summarized below.

Theorem 1. *For every field $\mathbb{F}$*

$$\mathrm{mSP}_{\mathbb{F}}(\text{st-con}) = \Omega(n^{1.5}).$$

Theorem 2. *For every finite field $\mathbb{F}$ and every linear secret sharing scheme over $\mathbb{F}$ realizing* st-con *the total number of bits in the shares is*

$$\Omega(n^{1.5} \log |\mathbb{F}|).$$

3 First Proof

3.1 Proof Outline

We use the following theorem of Gál [23] to prove our lower bound.

[1] The choice of the fixed nonzero vector $\boldsymbol{v}$ does not affect the size of the span program. It is always possible to replace $\boldsymbol{v}$ by another nonzero vector $\boldsymbol{v}'$ via a change of basis without changing the function computed and the size of the span program. Most often $\boldsymbol{v}$ is chosen to be the $\mathbf{1}$ vector (with all entries equal 1).

Theorem 3 ([23]). *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function. Let U denote the set of maxterms of f, and V denote the set minterms of f, and let $U' \subseteq U, V' \subseteq V$. If there exists a monotone span program of size s computing f over a field $\mathbb{F}$, then there exist matrices $F_1, \ldots, F_n$, each matrix has $|U'|$ rows and $|V'|$ columns (each row of the matrix is labeled by a $u \in U'$ and each column is labeled by a $v \in V'$) such that*

1. *$\sum_{i=1}^{n} F_i = \mathbf{1}$ (that is, the sum of the matrices over $\mathbb{F}$ is the all-one matrix).*
2. *The non-zero entries in F_i are only in rows labeled by a $u \in U'$ such that $u_i = 0$ and in columns labeled by a $v \in V'$ such that $v_i = 1$.*
3. *$\sum_{i=1}^{n} \mathrm{rank}_{\mathbb{F}}(F_i) \leq s$.*

In this section, we prove the result for GF(2), but the proof easily generalizes to other finite fields. The skeleton of the proof is as follows. We appropriately choose subsets U', V' of the maxterms and minterms of st-con. We show that for any matrices $F_1, \ldots, F_n$ satisfying (1) and (2) in Theorem 3, there exist "many" ($\Omega(n)$) matrices F_e, such that a large fraction ($\Omega(1)$) of the entries of F_e are zero entries. Also, every F_e has some "singleton" 1 entries at fixed positions, which are "well-spread" over the matrix. We then prove that every matrix F_e with "many" zero entries has rank $\Omega(n^{0.5})$, this proof uses the partial knowledge on the distribution of singletons, and the large number of zeros. By Theorem 3, this implies that the size of every monotone span program computing st-con over GF(2) has at least $\Omega(n^{0.5} \cdot n) = \Omega(n^{1.5})$ rows.

3.2 Details

To apply Theorem 3 we need to understand the minterms and maxterms of st-con. Every minterm of st-con is a simple directed paths from s to t. Every maxterm can be specified by a partition $S \cup T$ of V with $s \in S, t \in T$ where the edges in $S \times T$ are *excluded* and all other edges are included in the maxterm (that is, the maxterm contains all edges in $S \times S$, $T \times T$, and $T \times S$).

Defining U', V': Let $w = m/d$, where d is some constant to be fixed later.[2] We arrange the nodes of the graph in layers $L_0, L_1, \ldots, L_{d+1}$, where $L_0 = \{s\}, L_{d+1} = \{t\}$, and all other layers contain w nodes. We consider the restriction st-con$'$ of the st-con function to directed graphs that contain only edges directed from layer L_i to layer L_{i+1}. Note that the number of edges in the restricted function st-con$'$ is a constant fraction of the number of edges in the function st-con, so every lower bound for st-con$'$ implies the same lower bound for st-con (up to a constant factor). We define the subsets U', V' as follows. Let V' be all the s-t paths, that is, paths $s, v_1, \ldots, v_d, t$, where $v_i \in L_i$. Let U' be the set of all s-t cuts where 1/2 of the nodes in each layer L_i, where $1 < i < d$, are in S (and the other half is in T). Additionally, $\{s\} \cup L_1 \subset S$ and $\{t\} \cup L_d \subset T$. Note that $|V'| = w^d$ and $|U'| = \binom{w}{w/2}^{d-2}$.

[2] As we see later, $d = 4$ will do.

Assume there is a monotone span program over $\mathbb{F}$ computing st-con$'$ and let $F_1, \ldots, F_n$ be the matrices guaranteed by Theorem 3.[3] For an edge $e = (x, y)$, let R_e denote the restriction of F_e to rows labeled by a cut $u \in U'$ such that $u_e = 0$ (that is, the maxterm does not contain the edge (x, y)) and to columns labeled by a path $v \in V'$ such that $v_e = 1$ (that is, the path contains the edge (x, y)). Note that R_e has $w^{d-2} = |V'|/w^2$ columns and $0.25\binom{w}{w/2}^{d-2} = |U'|/4$ rows (as we consider cuts such that $x \in S$ and $y \in T$).[4] By (2) in Theorem 3, $\text{rank}_{\mathbb{F}}(R_e) = \text{rank}_{\mathbb{F}}(F_e)$. We say that R_e covers (u, v) if $u_e = 0$ and $v_e = 1$. Denote the set of edges e such that R_e covers (u, v) by $S(u, v)$.

We start with a few simple observations. Obseration 1 and Obseration 2 follow directly from (1) and (2) in Theorem 3 and the definition of the R_e's.

Observation 1. If $|S(u, v)| = 1$, then $F_e(u, v) = R_e(u, v) = 1$ for the edge $e \in S(u, v)$. We refer to such entries (u, v) as "singletons".

Observation 2. If $|S(u, v)|$ is even, then $R_e(u, v) = 0$ for some $e \in S(u, v)$.

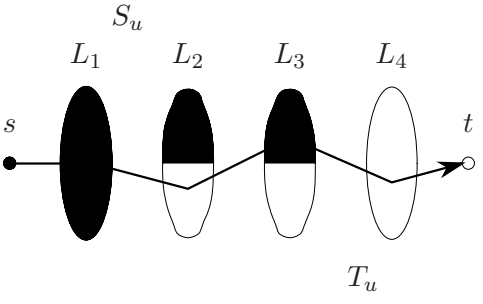


Fig. 1. An illustration of a path and a cut for which $|S(u, v)|$ is even. The vertices in S_u are black and the vertices in T_u are white. The edges in $S(u, v)$ are the edge between L_1 and L_2 and the edge between L_3 and L_4.

Lemma 1. *Let $c = 1/4$. For $d = 4$ there are at least $c|U'||V'|$ pairs (u, v) such that $|S(u, v)|$ is even.*[5]

Proof. From the definition of U', V', and (2) in Theorem 3, it follows that $S(u, v)$ is precisely the set of edges in v that belong to $S_u \times T_u$ (where the partition $S_u \cup T_u$ specifies the maxterm u). Fix some cut $u \in U'$. For a path v the size of $S(u, v)$ is even if the path has an even number of edges going from S_u to T_u. For $d = 4$ this is true if the vertex in L_2 is in T_u and the vertex in L_3 is in S_u, that

[3] For an edge $e = (s, x)$ or $e = (x, t)$, the matrix $F_e = \mathbf{0}$ (by the definition of the maxterms). We, therefore, ignore such matrices).

[4] This is true if $x \in L_j$ for $2 \leq j \leq d - 1$; the number of rows for $x \in L_1$ or $x \in L_{d-1}$ is $0.5\binom{w}{w/2}^{d-2} = |U'|/2$.

[5] For $d = 5$, the constant c is 0.5 and for any sufficiently large d, this constant approaches 1/2.

is, the edges in $S(u,v)$ are the edge between L_1 and L_2 and the edge between L_3 and L_4. See Fig. 1 for a description. Since half of the vertices of L_2 are in T_u and half of the vertices of L_3 are in S_u, for a quarter of the paths $v \in V'$, the size of $S(u,v)$ is even. □

We now move to our two main lemmas.

Lemma 2. *There exist at least cw^2 edges e such that R_e contains at least a $\frac{c}{d}$ fraction of zeros, where c is the constant from Lemma 1.*

Proof. We construct a set of edges as required, proceeding in iterations. By Lemma 1, for all (u,v) the set

$$B = \{(u,v) \,:\, |S(u,v)| \text{ is even}\}$$

must satisfy $R_e(u,v) = 0$ for some edge $e \in S(u,v)$. That is, we need to "cover" the set B by edges in this sense, where e covers $(u,v) \in B$ if $R_e(u,v) = 0$.

Denote by B_i the set of entries uncovered at the beginning of iteration i. In particular, $B_1 = B$. By Lemma 1, $|B_1| = c|U'||V'|$ for some constant c. We start an iteration i if $|B_i| \geq c|U'||V'|/2$. Since there are at most $w^2(d-1) - i \leq w^2 d$ edges to choose from, at least one of them should cover at least $\frac{c|U'||V'|/2}{w^2 d}$ uncovered entries (by the pigeon hole principle). We pick any of those e's and add it to the list. Note that the rectangle R_e has at most $|U'|/2 \cdot |V'|/w^2$ entries[6], thus a fraction of at least c/d of the entries of R_e are 0. Each selected edge e covers at most $|U'||V'|/2w^2$ uncovered entries (the number of entries in R_e). Since we halt only when at least $c|U'||V'|/2$ pairs in B have been covered, at least cw^2 iterations are needed. □

To complete the proof, it remains to show that every rectangle R_e with "many" zeros, as in Lemma 2, has high degree.

Lemma 3. *Let R_e, for $e = (x,y)$, be a rectangle with a fraction of at least c/d zero entries. Then* $\text{rank}_{\text{GF}(2)}(R_e) = \Omega(n^{0.5})$.

Proof. In the following proof we restrict our attention only to the rows and columns of R_e. First note that a fraction of at least $c/2d$ of the rows contain at a fraction of at least $c/2d$ zero entries (otherwise the fraction of zero entries in R_e is less than $c/2d \cdot 1 + (1 - c/2d) \cdot c/2d < c/d$). Thus, the number of rows with at least $c|V'|/(2w^2 d)$ zero entries is at least $c|U'|/(8d)$. We will show that these rows contain many distinct rows, which will imply that R_e has rank $\Omega(n^{0.5})$.

Fix any row u_0 of R_e with at least $c|V'|/(2w^2 d)$ zero entries. We show that the row u_0 can only appear in R_e a small number of times (labeled by different u's). Let M be the set of columns in which this row has zero entries; the size of M is at least $c|V'|/(4d)$. Let $e = (x,y)$, where x belongs to layer L_j for some j and $y \in L_{j+1}$.

We first prove that M contains a subset M' of paths of size $\epsilon \cdot w$ for some sufficiently small constant ϵ (to be fixed later) such that every two paths in M'

[6] This is the number of entries if $x \in L_1$, otherwise this number is $|U'|/4 \cdot |V'|/w^2$.

have no nodes in common except for x, y, s, t. Similarly to the proof of Lemma 2, we construct this set iteratively. In the first iteration, we add to M' some arbitrary path in M. We continue adding paths until there are ϵw paths. In iteration $i+1$, we have added i paths to M'. We prove that another path can be added so that all the paths in M' satisfy the invariant of being disjoint up to including s, x, y, t. Any path using one of the $w-i$ unused nodes in every layer L_k, where $k \neq j, j+1$, can be used here. The number of all columns of R_e with this property is at least $(w-i)^{d-2} \geq (w(1-\epsilon))^{d-2}$, thus the number of columns in R_e violating this property is at most

$$w^{d-2} - (w(1-\epsilon))^{d-2} = w^{d-2}(1-(1-\epsilon)^{d-2}) \approx w^{d-2}\epsilon(d-2) = |V'|\epsilon(d-2) < |V'|\epsilon d.$$

(for a sufficiently small constant ϵ). Taking $\epsilon \leq c/(4d^2)$, there are at least $c|V'|/(4d) - |V'|\epsilon d > 1$ paths in M satisfying this property.

Having proved $M' = \{v_1, \ldots, v_{\epsilon|M|}\}$ as above exists, we consider the set of rows

$$B' = \{u : e \notin u \text{ and } |S(u,v)| > 1 \text{ for every } v \in M'\}.$$

Notice that for every $u \notin B'$, where $e \notin u$, there exists a $v \in M'$ such that $|S(u,v)| = 1$, thus, by Obseration 1, $R_e(u,v) = 1$, however, $R_e(u_0, v) = 0$ since $v \in M$ (where M is the set of columns with zero entries in the row u_0). Thus, B' is the set of rows in R_e that can be equal to the row u_0. Recall that $e = (x,y) \in S(u,v)$ for every row u of R_e and every column v of R_e (by the definition of R_e). Thus, $S(u,v) > 1$ if the cut u does not contain at least one edge on the path v in addition to the edge (x,y).

We next show that B' is of negligible size. We do this by calculating the probability that a cut chosen with uniform distribution is B'. We choose a random cut $u = (S,T)$ by first choosing for each node in v_1 if its in S or in T, then the nodes corresponding to v_2, and so on, where the inclusion in S or T is selected at random according to the proportion of the remaining colors for that layer (conditioned on the choices for the previous v_i's). The cut u forms a singleton with a given v_i, selected in iterations i, if the node in v_i from L_k for $j' \leq j$ are in S, and the rest of the nodes in v_i are in T. This happens with probability at least $(1/2-\epsilon)^{d-2} \stackrel{\text{def}}{=} 1-f$. Thus, with probability at most f the cut u does not form a singleton with a given v_i. Note f is some constant. Therefore, $|S(u,v)| > 1$ for every $v \in M'$ with probability at most

$$f^{|M'|} = f^{\epsilon w} = 2^{-\theta(w)}.$$

This implies that the size of B' is at most $2^{-\theta(w)}|U'|/2$.

Since there are at least $c|U'|/(4d)$ rows with a fraction of at least $c/(2d)$ zeros, and each such row can appear at most $2^{-\theta(w)}|U'|/2$ in R_e, the number of distinct rows in R_e is at least

$$\frac{c|U'|/(4d)}{2^{-\theta(w)}|U'|/2} = 2^{\theta(w)}.$$

This implies that $\text{rank}_{\text{GF(2)}}(F_e) = \text{rank}_{\text{GF(2)}}(R_e) = \log(2^{\theta(w)}) = \theta(w) = \theta(n^{0.5})$. □

By Lemma 2 and Lemma 3, there are $\Omega(n)$ matrices F_e such that

$$\mathrm{rank}_{\mathrm{GF}(2)}(F_e) = \theta(n^{0.5}).$$

Thus, by Theorem 3, every monotone span program computing st-con has size $\Omega(n^{1.5})$.

4 Second Proof

In this proof we use a technique of [6] to prove lower bounds for monotone span programs. They prove that if the set of minterms of f contains a "big" set of self-avoiding minterms as defined below, then for every field $\mathbb{F}$ the size of every monotone span program over $\mathbb{F}$ computing f is "big".

Definition 2 (Self-Avoiding Minterms). *Let f be a monotone Boolean function and V be the set of all of its minterms. Let $V' \subseteq V$ be a subset of the minterms of f. We say that V' is* self avoiding for f*, if every $v \in V'$ contains a set $\mathrm{C}(v) \subseteq v$, called the core of v, such that the following three conditions are satisfied.*

1. $|\,\mathrm{C}(v)| \geq 2$.
2. *The set $\mathrm{C}(v)$ uniquely determines v in V'. That is, no other minterm in V' contains $\mathrm{C}(v)$.*
3. *For any subset $Y \subseteq \mathrm{C}(v)$, the set*

$$S_Y = \bigcup_{A \in V', A \cap Y \neq \emptyset} A \setminus Y$$

does not contain any minterm in V.

Note that (3) requires that S_Y contains no minterm from f, not just none from V'.

Theorem 4. *Let f be a monotone Boolean function, and let V' be a self-avoiding subset of minterms for f. Then for every field $\mathbb{F}$,*

$$\mathrm{mSP}_{\mathbb{F}}(f) \geq |V'|\,.$$

As in the first proof, we consider a graph with $m+2$ nodes, and let $w = m/4$. We arrange the nodes of the graph in layers $L_0, L_1, \ldots, L_5$, where $L_0 = \{s\}, L_5 = \{t\}$, and all other layers contain w nodes. We denote the nodes in layer L_j, where $1 \leq j \leq 4$ by $v_{j,1}, \ldots, v_{j,w}$. We consider the restriction st-con$'$ of the st-con function to directed graphs that contain only edges directed from layer L_i to layer L_{i+1}. We prove that every monotone span program for st-con$'$ has size $\Omega(w^3) = \Omega(n^{1.5})$. The proof is by exhibiting a self-avoiding set of minterms as defined in Definition 2.

The self-avoiding set for st-con′. For every $a, b, c \in \{1, \ldots, w\}$ there is a path $P_{a,b,c}$ in the set:

$$s, v_{1,a}, v_{2,b}, v_{3,c}, v_{4,a}, t.$$

That is, the indices of the nodes from L_1 and L_4 are equal. The core $\mathrm{C}(P_{a,b,c})$ is $\{(v_{1,a}, v_{2,b}), (v_{3,c}, v_{4,a})\}$. Clearly, the core determines the path $P_{a,b,c}$.

We have to show that for every $Y \subseteq \mathrm{C}(P)$ the set S_Y does not contain a path from s to t. If $|Y| = 1$ then S_Y does not contain an edge from one layer. E.g., if $Y = \{(v_{1,a}, v_{2,b})\}$ then S_Y does not contain any edges going from V_1 to V_2.

We next consider the somewhat more complex case when $|Y| = 2$. In this case S_Y is composed of the following edges:

1. $(s, v_{1,a})$ from the first level.
2. $(v_{1,a}, v_{2,b'})$ for every $b' \neq b$ from the second level.
3. $(v_{2,b}, v_{3,c'})$ for every c', and $(v_{2,b'}, v_{3,c})$ for every b' from the third level.
4. $(v_{3,c'}, v_{4,a})$ for every $c' \neq c$ from the fourth level.
5. $(v_{4,a}, t)$ from the fifth level.

Assume S_Y contains a path from s to t. Since $v_{2,b}$ does not have any incoming edges then the path has to pass through $v_{2,b'}$ for some $b' \neq b$. Thus it must pass through $v_{3,c}$. But $v_{3,c}$ has no outgoing edges in S_Y, contradiction.

To conclude, we have proved that st-con′ has a self-avoiding set of size $w^3 = O(n^{1.5})$ and Theorem 4 implies our main result – Theorem 1.

Acknowledgment. We would like to thank Eyal Kushilevitz for helpful discussions.

References

1. Ajtai, M., Fagin, R.: Reachability is harder for directed than for undirected finite graphs. J. Symb. Log. 55(1) (1990)
2. Babai, L., Gál, A., Wigderson, A.: Superpolynomial lower bounds for monotone span programs. Combinatorica 19(3), 301–319 (1999)
3. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. PhD thesis, Technion (1996), `www.cs.bgu.ac.il/beimel/pub.html`
4. Beimel, A., Chor, B.: Universally ideal secret sharing schemes. IEEE Trans. on Information Theory 40(3), 786–794 (1994)
5. Beimel, A., Gál, A.: On arithmetic branching programs. J. of Computer and System Sciences 59, 195–220 (1999)
6. Beimel, A., Gál, A., Paterson, M.: Lower bounds for monotone span programs. Computational Complexity 6(1), 29–45 (1997); Conference version: FOCS 1995
7. Beimel, A., Weinreb, E.: Separating the power of monotone span programs over different fields. SIAM J. on Computing 34(5), 1196–1215 (2005)
8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computations. In: Proc. of the 20th ACM Symp. on the Theory of Computing, pp. 1–10 (1988)
9. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, Heidelberg (1990)

10. Benaloh, J., Rudich, S.: Private communication (1989)
11. Blakley, G.R.: Safeguarding cryptographic keys. In: Merwin, R.E., Zanca, J.T., Smith, M. (eds.) Proc. of the 1979 AFIPS National Computer Conference, AFIPS Conference proceedings, vol. 48, pp. 313–317. AFIPS Press (1979)
12. Blundo, C., De Santis, A., de Simone, R., Vaccaro, U.: Tight bounds on the information rate of secret sharing schemes. Designs, Codes and Cryptography 11(2), 107–122 (1997)
13. Blundo, C., De Santis, A., Giorgio Gaggia, A., Vaccaro, U.: New bounds on the information rate of secret sharing schemes. IEEE Trans. on Information Theory 41(2), 549–553 (1995)
14. Brickell, E.F.: Some ideal secret sharing schemes. Journal of Combin. Math. and Combin. Comput. 6, 105–113 (1989)
15. Capocelli, R.M., De Santis, A., Gargano, L., Vaccaro, U.: On the size of shares for secret sharing schemes. J. of Cryptology 6(3), 157–168 (1993)
16. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: Proc. of the 20th ACM Symp. on the Theory of Computing, pp. 11–19 (1988)
17. Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000)
18. Csirmaz, L.: The size of a share must be large. In: De Santis, A. (ed.) Advances in Cryptology – EUROCRYPT 1994. LNCS, vol. 950, pp. 13–22. Springer, Heidelberg (1995); Journal version in J. of Cryptology 10(4), 223–231 (1997)
19. Csirmaz, L.: The dealer's random bits in perfect secret sharing schemes. Studia Sci. Math. Hungar. 32(3–4), 429–437 (1996)
20. Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)
21. van Dijk, M.: On the information rate of perfect secret sharing schemes. Designs, Codes and Cryptography 6, 143–169 (1995)
22. van Dijk, M.: A linear construction of secret sharing schemes. Designs, Codes and Cryptography 12(2), 161–201 (1997)
23. Gál, A.: A characterization of span program size and improved lower bounds for monotone span programs. Computational Complexity 10(4), 277–296 (2002)
24. Gál, A., Pudlák, P.: Monotone complexity and the rank of matrices. Inform. Process. Lett. 87, 321–326 (2003)
25. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proc. of the 13th ACM conference on Computer and communications security, pp. 89–98 (2006)
26. Ito, M., Saito, A., Nishizeki, T.: Secret sharing schemes realizing general access structure. In: Proc. of the IEEE Global Telecommunication Conf., Globecom 87, pp. 99–102 (1987); Journal version: Multiple assignment scheme for sharing secret. J. of Cryptology 6(1), 15–20 (1993)
27. Karchmer, M., Wigderson, A.: Monotone circuits for connectivity require super-logarithmic depth. SIAM J. on Discrete Mathematics 3(2), 255–265 (1990)
28. Karchmer, M., Wigderson, A.: On span programs. In: Proc. of the 8th IEEE Structure in Complexity Theory, pp. 102–111 (1993)
29. Karnin, E.D., Greene, J.W., Hellman, M.E.: On secret sharing systems. IEEE Trans. on Information Theory 29(1), 35–41 (1983)
30. Naor, M., Wool, A.: Access control and signatures via quorum secret sharing. IEEE Transactions on Parallel and Distributed Systems 9(1), 909–922 (1998)

31. Padró, C., Sáez, G.: Secret sharing schemes with bipartite access structure. IEEE Trans. on Information Theory 46, 2596–2605 (2000)
32. Rabin, M.O.: Randomized Byzantine generals. In: Proc. of the 24th IEEE Symp. on Foundations of Computer Science, pp. 403–409 (1983)
33. Reingold, O.: Undirected ST-connectivity in log-space. In: Proc. of the 37th ACM Symp. on the Theory of Computing, pp. 376–385 (2005)
34. Rónyai, L., Babai, L., Ganapathy, M.K.: On the number of zero-patterns of a sequence of polynomials. Journal of the AMS 14(3), 717–735 (2001)
35. Shamir, A.: How to share a secret. Communications of the ACM 22, 612–613 (1979)
36. Simmons, G.J., Jackson, W., Martin, K.M.: The geometry of shared secret schemes. Bulletin of the ICA 1, 71–88 (1991)
37. Stinson, D.R.: Decomposition construction for secret sharing schemes. IEEE Trans. on Information Theory 40(1), 118–125 (1994)
38. Tardos, E.: The gap between monotone and non-monotone circuit complexity is exponential. Combinatorica 8(1), 141–142 (1988)

Expressive Subgroup Signatures

Xavier Boyen[1] and Cécile Delerablée[2]

[1] Voltage, Palo Alto, California
xb@boyen.org
[2] Orange Labs - ENS
cecile.delerablee@orange-ftgroup.com

Abstract. In this work, we propose a new generalization of the notion of group signatures, that allows signers to cover the entire spectrum from complete disclosure to complete anonymity. Previous group signature constructions did not provide any disclosure capability, or at best a very limited one (such as subset membership). Our scheme offers a very powerful language for disclosing exactly in what capacity a subgroup of signers is making a signature on behalf of the group.

1 Introduction

Collective signatures allow an individual to make a signed statement anonymously on behalf of a coalition. Whereas ring signatures [30] are unconditionally anonymous, group signatures [17] come with an anti-abuse protection mechanism, whereby a tracing authority can lift a signature's anonymity to uncover the identity of the signer in case of necessity. In group signatures, membership to the group must be restricted and subject to a formal vetting and enrollment process of its members: these are desirable properties in many applications.

In many contexts, the blunt anonymity provided by group signatures goes too far, and it would be preferable to go half-way between full anonymity and full disclosure — e.g., to boost the credibility of a statement without completely engaging the individual responsibility of the signer. This is especially important in groups with many members, or with members of differing competences, or any time several signers wish to sign a joint statement while retaining some anonymity within a larger group.

The "Ad Hoc Group Signatures" from [20] at Eurocrypt 2004 provided a partial answer, by allowing a signer to disclose that he or she belongs to a particular subset of the group, not just the entire group. The "Mesh Signatures" from [11] at Eurocrypt 2007 went further by providing a very expressive language that signer(s) could use to make very specific statements as to the capacity in which they created the signature (such as, "undersigned by, either, five senators, or, two deputies and the prime minister"). Unfortunately mesh signatures were a generalization of ring signatures with no provision for traceability.

In this work, we propose a group signature with a mesh-like expressive language for proving and verifying complex propositions about group membership,

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 185–200, 2008.

including those whose fulfillment requires credentials from more than one group member. Given a valid signature, anyone can verify that it was created by a subgroup of signers that satisfied a certain condition (stated as an expression given with the signature), and learn nothing else. However, the tracing authority is able to unblind the signature and find out exactly how the condition was fulfilled, and thus who the signers are.

The construction we propose is based primarily on the mesh signatures of [11], which we modify in order to equip with a tracing capability. The tracing mechanism is inspired by a number of recent group signature constructions [12,1,13,21], which all made use of zero-knowledge proof systems in bilinear groups of composite order [10,22,23]. Compared to those, however, the present work provides a technical novelty: the composite algebraic group and the zero-knowledge proofs had to be flipped "inside out" in order to be compatible with the more expressive language that we implement.

Our signatures are efficient, both in the asymptotic and the practical sense: the signature size will be linear in the size of the expression that it represents, with a small proportionality constant. Although it would surely be nice to shrink the cryptographic part of the signature further down to a constant-size component, this is not of great importance here since the total signature size and verification time would still have to be linear or worse — because the verification expression has to be stated and used somewhere, and the access structure it represents is likely to change from one signature to the next. (Contrast this with regular group signatures, where it is more desirable to have signatures of constant size, because the group composition is fixed and need not be repeated.) For comparison, our fine-grained group signature is essentially as short and efficient as the mesh signature of [11], which is the most relevant benchmark for it is the only known anonymous signature that is as expressive as ours (albeit lacking the tracing capability).

Our scheme satisfies (suitable version of) the usual two main security properties of group signatures originally defined in [5]. The two properties are here: Full Anonymity for CPA-attacks [9] and Full Traceability with Dynamic Join [6], from which other natural requirements such as existential unforgeability, non-frameability, strong exculpability, collusion resistance, etc., can be shown to follow (see [5,6]). We shall define the two core properties precisely as they generalize to our more expressive notion of group signatures, and prove that our scheme satisfies them under previously analyzed complexity assumptions, in the standard model (unless a join protocol is used for strongly exculpability, in which case we need either random oracles or a common reference string to realize extractable commitments).

The name "Expressive Subgroup Signatures" is meant to capture that at the core these are group signatures, albeit not ones whose level of (revocable) anonymity is fixed and depends only on the group composition, but can be adjusted "in the field" with great precision, any time a new signature is created by any subgroup of signer(s) within the group boundaries.

1.1 Related Work

Ring signatures. Ring signatures were introduced in [30]. Each user in the system has a public key, and can generate a ring signature. Such a signature implicates some other users, chosen by the signer, and is such that a verifier is convinced that someone in the ring formed by the public keys of the signer and the chosen members is responsible for the signature, but nothing else. Constant-size ring signatures were constructed in [20]. Recently, a number of ring signatures without random oracles have been constructed from pairings, such as [18,7,31,11].

Mesh signatures. Mesh signatures were recently proposed [11] as a powerful generalization of ring signatures, with a rich language for striking the desired balance from full disclosure to full anonymity and almost anything in between (including complex statements involving, *inter alia*, trees of conjunctions and disjunctions of multiple potential signers). The work of [11] gave the first unconditionally anonymous ring signatures without random oracles as a special case.

Group signatures. Group signatures were first proposed in [17] to allow any member of a specific group to sign a message on behalf of the group, while keeping the signer anonymous within the group. However, a tracing authority has the ability to uncover the identity of the signer, which it should only do in certain extenuating circumstances. Group signatures have attracted much attention in the research community; we mention for example the works of [1,2,3,5,6,9,12,13,14,15,16,27,29,32].

For completeness, we mention the recently proposed notion of "attribute-based group signature" [26,25], which, contrarily to what the name might suggest, is a far cry from fulfilling our goal. (These signatures are rather inflexible, as they require that the attribute trees be constructed by the setup authority, and not the signer. Furthermore, verifying each attribute tree requires a different public key which must be requested *interactively* from the setup authority.)

2 Preliminaries

2.1 Composite-Order Pairings

Our construction makes use of bilinear pairings defined over groups of composite order, whose cryptographic applications were first investigated in [10].

A (symmetric) composite-order pairing is an efficiently computable function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are finite cyclic groups of order $N = pq$, and with the following properties:

Bilinearity: $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab \bmod N}$.
Non-degeneracy: $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order N and thus generates $\mathbb{G}_T$.

Although the composite group order N can be made public, it is usually important that the factorization $N = pq$ remains a secret. The most common hardness assumption that relies on hardness of factoring in the context of bilinear maps is called the Decision Subgroup assumption.

The Decision Subgroup Assumption. Let $\mathbb{G}$ be a bilinear group of order $N = pq$. Let $\mathbb{G}_p$ be the subgroup of points of order p with no residue of order q, that is, $u \in \mathbb{G}_p$ iff $u^p = 1 \in \mathbb{G}$. Similarly, we let $\mathbb{G}_q$ be the subgroup of points of order q congruent to 1 in $\mathbb{G}_p$.

Informally, the decision subgroup assumption says that one cannot efficiently distinguish $\mathbb{G}$ from $\mathbb{G}_p$ with non-negligible advantage.

Formally, we consider an "instance generator" $\mathcal{G}$, which, on input 1^λ, outputs a tuple $(p, q, \mathbb{G}, \mathbb{G}_T, e)$, where p and q are random λ-bit primes, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order $N = pq$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing. The subgroup decision problem is, given $(N, \mathbb{G}, \mathbb{G}_T, e)$ derived from an execution of $\mathcal{G}(1^\lambda)$, to decide whether a given element w was chosen randomly in $\mathbb{G}$ or in $\mathbb{G}_p$. The Subgroup Decision assumption states that this is infeasible in polynomial time with non-negligible probability above $1/2$, that of a random guess.

An alternative definition is to give the distinguisher two reference generators $g_N \in \mathbb{G}$ and $g_p \in \mathbb{G}_p$ in addition to $(N, \mathbb{G}, \mathbb{G}_T, e)$ and w; the task remains to decide whether $w \in \mathbb{G}$ or $w \in \mathbb{G}_p$. We use this definition to simplify our proofs.

The Poly-SDH Assumption. The traceability proof of the ESS scheme will be based on the unforgeability of the mesh signature scheme of [11], which was itself proved from the so-called Poly-SDH assumption in bilinear groups. The (q, ℓ)-Poly-SDH is a parametric assumption that mildly generalizes the earlier Strong Diffie-Hellman assumption [8] in such groups. It can be stated as:

(Poly-SDH) Given g, g^{α_1}, ..., $g^{\alpha_\ell} \in \mathbb{G}$ and $q\,\ell$ pairs $(w_{i,j}, g^{1/(\alpha_i + w_{i,j})})$ for $1 \le i \le \ell$ and $1 \le j \le q$, choose a list of values w_1, ..., $w_\ell \in \mathbb{F}_p$ and output ℓ pairs $(w_i, g^{r_i/(\alpha_i + w_i)})$ such that $\sum_{i=1}^{\ell} r_i = 1$.

2.2 Group Signatures

A group signature scheme involves a group manager, an opening manager, group members and outsiders. The group manager is able to add new members by issuing private keys thanks to a master key MK, while the opening manager can use the tracing key TK to revoke the anonymity of any group signature.
Such a scheme is specified as a tuple $\mathcal{GSS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Trace})$ of algorithms described as follows:

- The setup algorithm Setup generates, according to a security parameter, a public verification key PK, a master key MK, and a tracing key TK.
- The enrollment algorithm, Join, that generates a private signing key using the master key MK. The private key is then given to the new user.
- The group signing algorithm, Sign, that outputs a signature σ on a message M, using a group member's private key.
- The (usually deterministic) group signature verification algorithm, Verify, that takes as input the group verification key PK and a signature σ on a message M, and outputs either `valid` or `invalid`.
- The (usually deterministic) tracing algorithm, Trace, that takes a valid signature σ and a tracing key TK, and outputs the identity of a group member or $\perp$.

The following correctness and security properties are required.

Consistency. Given a group signature generated by a honest group member, the verify algorithm should output `valid`, and the tracing algorithm should identify the actual signer.

Security. Bellare, Micciancio, and Warinschi [5] characterize the fundamental properties of group signatures in terms of two crucial security properties from which a number of other properties follow. The two important properties are *Full Anonymity* and *Full Traceability*.

It is noted in [5] that the full traceability property implies that of *exculpability* [4], which is the requirement that no party should be able to frame a honest group member as the signer of a signature he did not make, not even the group manager. However, the model of [5] does not consider the possibility of a (long-lived) group master, which leaves it as a potential framer. To address this problem and achieve the notion of *strong exculpability*, introduced in [2] and formalized in [29,6], one also needs an interactive enrollment protocol, call *Join*, at the end of which only the user himself knows his full private key; the same mechanism may also enable concurrent dynamic group enrollment [6,29]. In this work, we opt for this stronger notion and thus we shall explicitly describe such a Join protocol.

We refer the reader mainly to [5] for more precise definitions of these and related notions.

2.3 Mesh Signatures

We now briefly recapitulate the notion of mesh signature introduced in [11].

In short, a mesh signature is a non-interactive witness-indistinguishable proof that some monotone boolean expression $\Upsilon(L_1, \ldots, L_n)$ is true, where the input literals L_i denote the validity of "atomic signatures" of the form $\{Msg_i\}_{Key_i}$ for arbitrary messages and different keys. (The special case of ring signatures [30] corresponds to Υ being a flat disjunction.)

Admissible mesh expressions Υ consist of trees of And, Or, and Threshold gates, and single-use input literals, generated by the following grammar:

$$
\begin{array}{lll}
\text{EXPR} ::= L_1 \mid \ldots \mid L_\ell & & \text{single-use input symbols} \\
\quad \mid \geq_t \{\text{EXPR}_1, \ldots, \text{EXPR}_m\} & & t\text{-out-of-}m \text{ threshold, with } 1 < t < m \\
\quad \mid \wedge \{\text{EXPR}_1, \ldots, \text{EXPR}_m\} & & m\text{-wise conjunction, with } 1 < m \\
\quad \mid \vee \{\text{EXPR}_1, \ldots, \text{EXPR}_m\} & & m\text{-wise disjunction, with } 1 < m
\end{array}
$$

Not all literals need to be true in order for Υ to be satisfied. However the mesh signature must only attest to the truth of Υ without revealing how it is satisfied: this is the *anonymity* property. Conversely, a signer should not be able to create a mesh signature without possessing a valid atomic signature for every literal set to true: this is the *unforgeability* property.

2.4 Security of Expressive Subgroup Signatures

ESS are just as expressive as mesh signatures, and provide the same anonymity, except that the latter can be lifted by a tracing authority. We require:

ESS Anonymity. The notion of anonymity we seek is not that we wish to hide the identity of the users named in the ESS expression Υ (which is public anyway), but to hide who among the users actually created the signature.

Precisely, we require that the identity of the actual signer(s) be computationally indistinguishable in the set of all valid ESS signatures specified by the same expression Υ, even under full exposure of all user keys. This is the same notion as in the mesh signatures of [11], except that here the requirement is computational and not information-theoretic in order not to stymie the tracing authority, and is of course conditional on the secrecy of the tracing key.

ESS Traceability. Traceability is the group-signature version of unforgeability. For ESS, as for mesh signatures before them, this notion is tricky to formalize because of the potentially complex dependences that may exist between good and forged signatures. To see this, consider a coalition of two forgers, U_1, U_2, and a honest user, U_3. Suppose that the forgers fabricate a valid ESS signature σ for the expression $\Upsilon = \{m_1\}_{U_1} \vee (\{m_2\}_{U_2} \wedge \{m_3\}_{U_3})$, that can be traced the subgroup U_2, U_3. Is that a successful forgery? What if σ traced to U_2 only?

The answer is a double "yes": in the first case, because U_3 was wrongly designated by the tracer; and in the second case, because U_2 alone could not have satisfied Υ, which means that some guilty party escaped detection. If on the other hand, the finger were pointed at U_1, the signature would have a satisfactory explaination that involves only (the parties that we know to be) the culprits: this would be a failed forgery since the coalition got caught.

The ESS traceability challenge is thus, for any coalition of signers, to come up with a valid signature σ for an expression $\Upsilon(L_1, \ldots, L_n)$, such that σ, either, traces to at least one user outside of the coalition, or, traces to a subset of the coalition that does not validate Υ (that is, when Υ is "false" after setting the designated literals L_i to "true" and only those).

Strong Exculpability. This last notion is borrowed straight from group signatures [2,29,6], and is orthogonal to the above. It gives any user the possibility to dispute his alleged binding to any certificate that he did not request. To defend from such accusation, the group manager (who signed the disputed certificate) must produce a valid certificate request signed by the user's individual key registered in some PKI. The enrollment protocol must guarantee that only the user learns the private key associated with their certificates. Together, this prevents the ESS group manager from framing users for signatures they did not make.

2.5 Formal Security Models

We now specify the formal ESS security model in accordance with the previously stated requirements.

Anonymity. The ESS anonymity game is played between a challenger and an adversary.

Initialization. The challenger gives to the adversary the public parameters of an ESS.

Interaction. The following occurs interactively, in any order, driven by the adversary.

User enrollment. The adversary may ask the challenger to enroll a polynomially bounded number of new users in the group. The adversary may either impersonate the user in the enrollemnt protocol, or ask the challenger to simulate it all by itself. The resulting public certificates are published.

Signature queries. The adversary may ask the challenger to sign any ESS expression Υ on behalf of the users it controls.

Key recovery. The adversary may ask the challenger to reveal the group signing key of any user.

The challenger processes each request before accepting the next one.

Challenge: The adversary finally output a specification Υ and two sets of assignments χ_1 and χ_2 to its literals L_i, that both cause Υ to be satisfied. These truth assignments indicate which users are supposed to sign Υ. The adversary must also supply the necessary atomic signatures for the users for which it has the keys.

The challenger chooses one assignment $b \in \{1, 2\}$ at random, and creates an ESS signature σ on the specification Υ using only atomic signatures corresponding to the true literals in χ_i. The signature σ is given to the adversary.

Guess: The adversary makes a guess b', and wins the game if $b = b'$.

The adversary's advantage in the ESS anonymity game is $\Pr[b = b'] - 1/2$, where the probability is defined over the random coins used by all the parties.

Traceability. The ESS traceability game is played between a challenger and an adversary.

Initialization. The challenger gives to the adversary the public parameters of an ESS. The challenger also reserves a number ℓ of user indices to represent the "honest users" under its control.

Interaction. The following occurs interactively, in any order, driven by the adversary.

Honest user enrollment. The adversary may request that the challenger create up to ℓ honest users, kept in the challenger's control. The challenger publishes the corresponding certificates.

Corrupted user enrollment. The adversary makes polynomially user enrollment queries, for the users under the adversary's control. The adversary chooses or receives the user secret keys in accordance with the chosen enrollment protocol. The challenger computes the corresponding certificates in accordance with the enrollment protocol, and publishes them.

ESS signature queries. The adversary makes up to q ESS signature queries, one at a time, on specifications Υ_j, indicating to the challenger which ones of the honest users are supposed to issue the signature. To be acceptable, each request must be for

a signature that the specified subset of honest users is supposed to be able to make based on the specification and supporting atomic signatures provided by the adversary.
The adversary may also make up to q queries for atomic signatures, to each of the users controlled by the challenger.

The challenger processes each request before accepting the next one.

Forgery: The adversary finally output a fresh valid ESS signature σ for some specification Υ of its choice. It wins the game if the list of literals L_i designated by the tracing algorithm on input σ fails to satisfy the two following properties:

1. All the designated literals L_i correspond to atomic signatures $\{Msg_i\}_{User_i}$ under the adversary's control (either because the adversary controls the corresponding user, or had obtained the atomic signature by querying the challenger).
2. The specification formula $\Upsilon(..., L_i, ...)$ can be satisfied by setting all the designated literals to "true" ($\top$) and all the other literals to "false" ($\bot$).

The adversary's advantage in the ESS traceability game is simply the probability that it wins the game. The probability is defined over the random coins used by all the parties.

3 Construction

Our Expressive Subgroup Signature construction will bring together a number of different techniques.

To get the anonymity properties we seek, we will naturally start with the ring/mesh signatures from [11], which comes with a powerful language and proof system. We use it to create an anonymous group identification mechanism for certificates issued by the group manager. Since we need a signature scheme and not just an identification scheme, we shall extend the certificates into certificate chains ending with actual signatures from users' keys. This part is easy to do using the mesh language, so this step will be a simple matter of specifying how the various terminal signatures and their supporting certificates should be assembled. This gives us a multi-user anonymous signature with a central authority. However, we still lack traceability.

To get traceability, we need to build a trapdoor that will remove the blinding from the mesh signatures. Recall that the ring and mesh signatures from [11] consist of one signature element per ring or mesh member. Some of those elements are "live", meaning that they were created using the member's actual secret key. The remaining elements are "blank" and do not contribute to the verification equation. Since it would be easy to tell who the signers were just by finding the live elements, the elements are information-theoretically blinded so that they all look the same. Here, to get traceability, we shall swap out the perfect blinding for one that has a trapdoor. Since the mesh signatures require a bilinear pairing

for their verification, we shall add the trapdoor into the bilinear group, using a standard trick used in several previous constructions [22]. We simply translate the signatures into a bilinear group of composite order, and restrict the blinding elements to one of its two algebraic subgroups. An adversary will just see smoke under the proper hardness assumption [10]; but a tracing authority that knows the order of the subgroups will be able to cancel the blinding (by pairing each signature component with a neutral element in that subgroup), and hence distinguish which signature components are live and which ones are blank.

In the following subsections, we explain step-by-step how to construct expressive subgroup signatures. We work in an algebraic group $\mathbb{G}$ of composite order $N = pq$ and equipped with a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We call $\mathbb{G}$ the bilinear group and $\mathbb{G}_T$ the targer group; both are of order N. Bilinearity is the property that $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab \bmod N}$.

3.1 User Credentials

Users must be affiliated with the group in order to create signatures, which means that they must have acquired proper credentials from the group manager (which controls the user vetting and enrollment process).

In its most basic instantiation, a certificate for user i could simply be a secret key for the Boneh-Boyen signature scheme [8]. The secret key $(y_i, z_i) \in \mathbb{Z}_p^2$ would be securely handed over to the user, and the corresponding public key $(g^{y_i}, g^{z_i}) \in \mathbb{G}^2$ signed by the group manager and perhaps published as part of the group description. A signature on $m \in \mathbb{Z}_p$ would be a random pair $(t, S) \in \mathbb{Z}_p \times \mathbb{G}$, where $S = g^{1/(y_i + m + z_i t)}$, which is verifiable by testing $e(S, g^{y_i} g^m g^{z_i t}) = e(g, g)$. The drawback is that the group manager would know y_i and z_i and would thus be able to create signatures on the user's behalf. We would also need to embed a tracing trapdoor into all user-generated signatures.

In the preferred instantiation, a group certificate will depend on a secret component that is known only to the user, to prevent users from being framed. It should also depend on a secret from the manager, to guarantee traceability. Using a technique close to the one proposed by Delerablée and Pointcheval [19], we let the credentials for user i consist of a secret key (x_i, y_i, z_i) and a public certificate (A_i, B_i, C_i), where $A_i = g^{y_i/(\gamma + x_i)}$, $B_i = g^{1/(\gamma + x_i)}$, and $C_i = g^{z_i/(\gamma + x_i)}$, for some random x_i. Here, γ and $\Gamma = h^\gamma$ are respectively the secret and public key of the group manager, and g and h are two fixed generators of the group $\mathbb{G}$. The certificate of user i is the triple (A_i, B_i, C_i) signed by the group manager. For randomly chosen $t \in \mathbb{Z}_p$, an "atomic signature" on $m \in \mathbb{Z}_p$ will be a pair:

$$\sigma = (t, S) \in \mathbb{Z}_p \times \mathbb{G} \qquad \text{s.t.} \qquad S = (\Gamma h^{x_i})^{1/(y_i + m + z_i t)} \ .$$

The verification equation is thus: $e(S, A_i B_i^m C_i^t) = e(h, g)$.

For simplicity reasons, we can merely suppose a enrollment protocol where the user chooses (y_i, z_i), sends (g^{y_i}, g^{z_i}) to the group manager along with a proof of knowledge, and receives (x_i, A_i, B_i, C_i) in return. Nevertheless, following

a technique from [19], in Section 3.7 we present a more complex "dynamic" enrollment protocol, which renders our scheme secure under concurrent join [28], and provides strong user exculpability [6] against dishonest managers.

3.2 Atomic Signatures

Using their credentials, users are able to create atomic signatures on any message of their choice, which for simplicity we assume represented as an integer $m \in \mathbb{Z}_p$. Atomic signatures provide no anonymity; they merely serve as building blocks in more complex assemblies.

An atomic signature created from credentials as above is a pair $(t, S) \in \mathbb{Z}_p \times \mathbb{G}$ that satisfies a verification equation of the form,

$$e(S, \underbrace{A_i B_i^m C_i^t}_{R}) = e(h, g) \ ,$$

with respect to a publicly verifiable certificate (A_i, B_i, C_i) associated to user i. We observe for later use that this is exactly a Boneh-Boyen signature, and that the right-hand side $e(h, g)$ in the verification equation is the same for all users.

3.3 Ring Signatures

Once we have atomic signatures of the previous form, we can easily construct an information-theoretically anonymous ring signature, based on the approach proposed in [11]. Suppose that there are n users with public certificates (A_1, B_1, C_1) through (A_n, B_n, C_n), and consider the following verification equation for some message m, or more generally, for respective user messages m_1 through m_n:

$$\prod_{i=1}^{n} e(S_i, \underbrace{A_i B_i^{m_i} C_i^{t_i}}_{R_i}) = e(h, g) \ .$$

Any one of the n users is able to create, by himself, a vector of n pseudo-signatures (t_i, S_i) for $i = 1, \ldots, n$ that will jointly verify the preceding equation. In order to do so, the user will need his own key and everyone else's certificates. For example, user 1 would pick random $r_2, \ldots, r_n$, and $t_1, \ldots, t_n$, and set:

$$S_1 = (\Gamma h^{x_1})^{1/(y_1 + m_1 + z_1 t_1)} \cdot \left[\prod_{i=2}^{n} R_i^{r_i} \right] , \quad S_2 = \left[R_1^{-r_2} \right] , \quad \ldots \ , \quad S_n = \left[R_1^{-r_n} \right] .$$

It is easy to see that, for any random choice of $r_i \in \mathbb{Z}_p$, the blinding terms in the square brackets will cancel each other in the product of pairings in the verification equation; *e.g.*, $e(R_2^{r_2}, R_1)$ from S_1 will cancel $e(R_1^{-r_2}, R_2)$ from S_2. What is left is the Boneh-Boyen signature component $(\Gamma h^{x_1})^{1/(y_1 + m_1 + z_1 t_1)}$ in S_1, which in the verification equation will produce the value $e(h, g)$ we seek.

For the example of user 1 being the actual signer, the cancellation that occurs is, *in extenso*, if we let $S'_1 = (\Gamma h^{x_1})^{1/(y_1+m_1+z_1 t_1)}$:

$$\begin{aligned}\prod_{i=1}^{n} e(S_i, R_i) &= e(S_1, R_1) \cdot e(\prod_{i=2}^{n} R_i^{r_i}, R_1) \cdot \prod_{i=2}^{n} e(S_i, R_i) \\ &= e(S'_1, R_1) \cdot e(\prod_{i=2}^{n} R_i^{r_i}, R_1) \cdot \prod_{i=2}^{n} e(R_1^{-r_i}, R_i) \\ &= e(h, g) \cdot \prod_{i=2}^{n} e(R_i, R_1)^{r_i} \cdot \prod_{i=2}^{n} e(R_1, R_i)^{-r_i} \quad = e(g, h)\end{aligned}$$

The point is that user 2 (or any other user j) could have achieved the same result by using his own secret key inside S_2 (or S_j), but nobody else could, without one of the users' key. Also, because there are $2n$ components in the signature, but $2n-1$ randomization parameters and 1 perfectly symmetric verification equation, it is easy to see that the joint distribution of the full signature $(t_i, S_i)_{i=1}^n$ is the same regardless of which one of the n listed users created it.

Hence, this is a ring signature, *i.e.*, a witness-indistinguishable proof for the disjunction "$\{m_1\}_{\text{user}_1} \vee \{m_2\}_{\text{user}_2} \vee \ldots \vee \{m_n\}_{\text{user}_n}$". The signature can be shown to be unconditionally anonymous, and existentially unforgeable under the n-Poly-SDH assumption [11], which slightly generalizes the SDH assumption [9].

3.4 Mesh Signatures

The next step is to turn those ring signatures into something that is much more expressive. Recall that ring signatures can be viewed as witness-indistinguishable "disjunctions" of individual signatures. Since the disjunction $L_1 \vee L_2 \vee \ldots \vee L_n$ is the least restrictive of all (non-trivial) propositional expressions over $L_1, \ldots, L_n$, it should be possible to express different statements by adding more constraints to the signature. *E.g.*, we could require that supplemental verification equations be satisfied conjointly. The "mesh signatures" of [11] are based on this principle.

A classic result from [24] shows that any monotone propositional expression over a set of literals can be represented efficiently and deterministically using a system of linear equations $\{\sum_{i=1}^{n} \lambda_{i,j}\nu_i = \lambda_j\}_{j=1}^{k+1}$ over the same number of variables: a literal L_i will be true in a true assignment if and only if the corresponding variable ν_i has a non-zero value in the corresponding system solution (of which there may be many).

In the construction of [11], the linear system coefficients $\lambda_{i,j}$ will become public exponents in the verification equations. Depending on the expression it represents, a mesh signature $(t_i, S_i)_{i=1}^n$ requires $1 \leq k+1 \leq n$ verification equations (with the usual $R_i = A_i B_i^{m_i} C_i^{t_i}$ computable from public values):

$$\prod_{i=1}^{n} e(S_i, R_i) = e(h, g) \ ,$$
$$\prod_{i=1}^{n} e(S_i, R_i)^{\lambda_{i,1}} = 1 \ ,$$
$$\cdots$$
$$\prod_{i=1}^{n} e(S_i, R_i)^{\lambda_{i,k}} = 1 \ .$$

To make a signature, the signer, or coalition of signers, must prove that the propositional expression has a solution, *i.e.*, that there is a vector of S_i that passes all the equations. This can be done by setting $S_i \leftarrow ((\Gamma h^{x_i})^{1/(y_1+m_i+z_1t_1)})^{\nu_i}$ given any solution vector $(\nu_1, \ldots, \nu_n)$ of the linear system. However, for every index i with a non-zero solution $\nu_i \neq 0$, the signer(s) will be unable to create S_i unless they possess or are able to create the atomic signature $(\Gamma h^{x_i})^{1/(y_1+m+z_1t_1)}$. Only for $\nu_i = 0$ can they get by without it.

This procedure results in a valid signature, but not in an anonymous one. The last step is thus to hide the witness, *i.e.*, the vector $(\nu_1, \ldots, \nu_n)$ used to build the S_i. This is done by adding blinding terms to the S_i just as in the ring signature. The result is an unconditionally anonymous signature for arbitrary monotone propositional expressions.

The entire mesh signing process and its security proofs are somewhat more involved. Full mesh signatures also require a presence of a dummy user "in the sky" (with a public random public key and no known secret key), who will "sign" a hash of the entire mesh expression in order to "seal" it. We refer the reader to [11] for details.

3.5 Tracing Trapdoor

We now have an expressive anonymous signature, albeit not a traceable one. To make mesh signatures traceable, we need to redefine the mesh signature scheme in bilinear groups of composite order N. The factorization $N = pq$ is a trapdoor that is only known to the tracing authority. Let thus $\mathbb{G}_N \simeq \mathbb{G}_p \otimes \mathbb{G}_q$.

ESS signatures are defined as mesh signatures in a composite-order group $\mathbb{G}$. We do require however that the "main" generator g generate only the subgroup $\mathbb{G}_p$ of order p. That is, we impose that $g^p = 1 \in \mathbb{G}$ or equivalently $g \equiv 1 \in \mathbb{G}_q$. Since the A_i, B_i, C_i, and thus the R_i, are powers of g, all those elements will belong in the subgroup $\mathbb{G}_p$ of order p. It is easy to see that, since the verification equation is of the form $\prod e(S_i, R_i) = e(h, g)$, both sides will always evaluate into the target subgroup of order p, with no contribution of order q. It follows that only the $\mathbb{G}_p$ components of the S_i will matter for ESS verification.

In order to provide a tracing capability, we pick h as a generator of the entire group $\mathbb{G}$, hence with a non-trivial component $h \not\equiv 1 \in \mathbb{G}_q$. The same will be true for the public key $\Gamma = h^\gamma$. As a result, all the user-created atomic signatures of the form $S = (\Gamma h^{x_i})^{1/(\cdots)}$ will also contain a non-trivial component $S \not\equiv 1 \in \mathbb{G}_q$,

which has no effect on the ESS verification equation, per the preceding argument. These order-q components will be our tracers, since they appear in all atomic signatures (which are powers of $h \in \mathbb{G}$), but not in any of the blinding coefficients (which are powers of $g \in \mathbb{G}_p$).

Since we now work in a composite-order group of order N, we redefine the user's signing exponents in $\mathbb{Z}_N$ instead of $\mathbb{Z}_p$.

Remark that if h had no residue of order q, then everything would be in $\mathbb{G}_p$. It would be as if the subgroup $\mathbb{G}_q$ did not exist, and the ESS scheme would reduce to an information-theoretically untraceable mesh signature in $\mathbb{G}_p$. As the Decision Subgroup assumption [10] states that $h \in \mathbb{G}$ and $h \in \mathbb{G}_p$ should look the same to an outsider, it follows that our tracing mechanism cannot be public and will thus require some trapdoor (in this case, the factorization of N).

3.6 Tracing Procedure

The presence of a non-trivial residue of order q in h will act as a silent tracer for lifting the anonymity of any signature, using the factorization of N as trapdoor.

To unblind an ESS signature $(t_i, S_i)_{i=1}^n$, the tracing authority raises each S_i to the power of p, to strip it from all components of order p. Then, for each i:

- If the residue $(S_i)^p = 1$, there was no contribution from h in S_i, hence $\nu_i = 0$, and thus the truth value of the associated literal L_i is "false". Conclusion: user i did not participate in the creation of the ESS signature.
- If the residue $(S_i)^p \neq 1$, there was some h contribution in S_i, hence $\nu_i \neq 0$, and thus the truth value of the associated literal L_i is "true". Conclusion: (an atomic signature issued by) user i took part in the ESS signature.

The tracer can thus efficiently determine the exact set of users that are involved (and in what capacity).

We emphasize that, unlike tracing schemes in many other contexts that can only guarantee that one of the guilty parties will be exposed, here the tracing authority finds out exactly how the signature was constructed, and thus uncovers the identity of *all* of the culprits.

Notice also that such detailed "exhaustive tracing" requires signatures whose size is (at least) linear in the size of the propositional expression. Hence, in that respect, our scheme is optimally compact up to a constant factor.

3.7 Concurrent Join Protocol

As in [19] we can define a Join protocol, using some standard techniques: an extractable commitment (Ext-Commit), a zero-knowledge proof of equality of the discrete logarithms (NIZKPEqDL), and a zero-knowledge proof of knowledge of a discrete logarithm (NIZKPoKDL). During this protocol, a future group member ($\mathcal{U}_i$) interacts with the group manager ($\mathcal{GM}$), in order to obtain a valid group certificate (A_i, B_i, C_i), with a private key (x_i, y_i, z_i), with (y_i, z_i) not known

by the group manager. We suppose, as in [19] that there is a separated PKI environment: each user $\mathcal{U}_i$ has a personal secret key $\mathsf{usk}[i]$ and the corresponding certified public key $\mathsf{upk}[i]$.

We refer to the full version of the paper for the details of the Join protocol.

3.8 The Full ESS Construction

The step-by-step construction outlined above gives us the complete ESS scheme. The only operational differences with the mesh signature scheme of [11] are:

1. the setup, which asks for a bilinear group $\mathbb{G}$ of composite order $N = pq$, two generators $g \in \mathbb{G}_p$ and $h \in \mathbb{G}$, and a group manager's public key $\Gamma = h^\gamma$;
2. the existence of two additional algorithms or protocols: one for joining the group, the other for tracing a signature.

For reference purposes, the complete ESS construction in full detail as well as security proofs can be found in the full version of the paper. The construction follows exactly the outline given above. Most of the technicalities are borrowed directly from the mesh signature scheme of [11], with which the ESS scheme shares many similarities.

4 Security

Theorem 1 (Anonymity). *There is no PPT algorithm that wins the Expressive Subgroup Signature anonymity game over $\mathbb{G}$ with advantage ϵ in time τ, unless the Decision Subgroup problem in $\mathbb{G}$ is decidable in time $\tau' \approx \tau$ with advantage $\epsilon' \geq \epsilon/2$.*

Theorem 2 (Traceability). *There is no PPT algorithm that wins the Expressive Subgroup Signature traceability game over $\mathbb{G}$ with advantage ϵ in time τ, unless the Decision Subgroup problem is decidable in time τ' with advantage ϵ', and mesh signatures in $\mathbb{G}_p$ can be existentially forged in time τ'' with advantage ϵ'', where $\tau' + \tau'' \approx \tau$ and $\epsilon' + \epsilon'' \geq \epsilon/2$.*

5 Conclusion

In this work, we have proposed a new generalization of the notion of group signatures, that allows signers to cover the entire spectrum from complete disclosure to complete anonymity. Previous group signature constructions did not provide any disclosure capability, or at best a very limited one (such as subset membership). Our scheme offers a very powerful language for disclosing exaclty in what capacity a subgroup of signers is making a signature on behalf of the group.

References

1. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005), http://eprint.iacr.org/
2. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)
3. Ateniese, G., Song, D.X., Tsudik, G.: Quasi-efficient revocation in group signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 183–197. Springer, Heidelberg (2003)
4. Ateniese, G., Tsudik, G.: Some open issues and new directions in group signatures. In: Franklin, M. (ed.) FC 1999. LNCS, vol. 1648, pp. 196–211. Springer, Heidelberg (1999)
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
6. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)
7. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
8. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
9. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
10. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
11. Boyen, X.: Mesh signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 210–227. Springer, Heidelberg (2007)
12. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
13. Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
14. Camenisch, J.: Efficient and generalized group signatures. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 465–479. Springer, Heidelberg (1997)
15. Camenisch, J., Groth, J.: Group signatures: Better efficiency and new theoretical aspects. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 120–133. Springer, Heidelberg (2005)
16. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
17. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

18. Chow, S.M., Wei, V.K.-W., Liu, J.K., Yuen, T.H.: Ring signatures without random oracles. In: ASIACCS 2006 Conference on Computer and Communications Security, Taipei, Taiwan, pp. 297–302. ACM Press, New York (2006)
19. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006)
20. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
21. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
22. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
23. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155 (2007), `http://eprint.iacr.org/`
24. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of Structures in Complexity Theory, pp. 102–111 (1993)
25. Khader, D.: Attribute based group signature with revocation. Cryptology ePrint Archive, Report 2007/241 (2007), `http://eprint.iacr.org/`
26. Khader, D.: Attribute based group signatures. Cryptology ePrint Archive, Report 2007/159 (2007), `http://eprint.iacr.org/`
27. Kiayias, A., Yung, M.: Extracting group signatures from traitor tracing schemes. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 630–648. Springer, Heidelberg (2003)
28. Kiayias, A., Yung, M.: Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076 (2004), `http://eprint.iacr.org/`
29. Kiayias, A., Yung, M.: Group signatures with efficient concurrent join. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 198–214. Springer, Heidelberg (2005)
30. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
31. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 166–180. Springer, Heidelberg (2007)
32. Song, D.X.: Practical forward secure group signature schemes. In: ACM CCS 2001 8th Conference on Computer and Communications Security, Philadelphia, PA, USA, November 5–8, 2001, pp. 225–234. ACM Press, New York (2001)

Anonymous Proxy Signatures

Georg Fuchsbauer and David Pointcheval

École normale supérieure, LIENS - CNRS - INRIA, Paris, France
http://www.di.ens.fr/{~fuchsbau,~pointche}

Abstract. We define a general model for consecutive delegations of signing rights with the following properties: The delegatee actually signing and all intermediate delegators remain anonymous. As for group signatures, in case of misuse, a special authority can *open* signatures to reveal the chain of delegations and the signer's identity. The scheme satisfies a strong notion of non-frameability generalizing the one for dynamic group signatures. We give formal definitions of security and show them to be satisfiable by constructing an instantiation proven secure under general assumptions in the standard model. Our primitive is a proper generalization of both group signatures and proxy signatures and can be regarded as non-frameable dynamic hierarchical group signatures.

1 Introduction

The concept of delegating signing rights for digital signatures is a well studied subject in cryptography. The most basic concept is that of proxy signatures, introduced by Mambo et al. [MUO96] and group signatures, introduced by Chaum and van Heyst [CvH91]. In the first, a *delegator* transfers the right to sign on his behalf to a *proxy signer* in a *delegation protocol*. Now the latter can produce *proxy signatures* that are verifiable under the delegator's public key. Security of such a scheme amounts to unforgeability of proxy signatures, in that an adversary cannot create a signature without having been delegated, nor impersonate an honest proxy signer.

On the other hand, in a group signature scheme, an authority called the *issuer* distributes signing keys to *group members*, who can then sign on behalf of the group, which can be viewed as delegating the group's signing rights to its members—there is one single *group signature verification key*. The central feature is anonymity, meaning that from a signature one cannot tell which one of the group members actually signed. In contrast to ring signatures [RST01], to preclude misuse, there is another authority holding an *opening key* by which anonymity of the signer can be revoked. Generally, one distinguishes *static* and *dynamic* groups, depending on whether the system and the group of signers are set up once and for all or members can join dynamically. For the dynamic case, a strong security notion called *non-frameability* is conceivable: Nobody—not even the issuer nor the opener—is able to produce a signature that opens to a member who did not sign. The two other requirements are *traceability* (every valid signature can be traced to its signer) and *anonymity*, that is, no one except the opener can distinguish signatures of different users.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 201–217, 2008.

It is of central interest in cryptography to provide formal definitions of primitives and rigorously define the notions of security they should achieve. Only then can one *prove* instantiations of the primitive to be secure. Security of group signatures was first formalized by Bellare et al. [BMW03] and then extended to dynamic groups in [BSZ05]. The model of proxy signatures and their security were formalized by Boldyreva et al. [BPW03].[1]

The main result of this paper is to unify the two above-mentioned seemingly rather different concepts, establishing a general model which encompasses proxy and group signatures. We give security notions which imply the formal ones for both primitives. Moreover, we consider consecutive delegations where *all* delegators (except the first of course) remain anonymous. As for dynamic group signatures, we define an opening authority separated from the issuer and which in addition might even be different for each user (for proxy signatures, a plausible setting would be to enable the users to open signatures on their behalf). We call our primitive *anonymous proxy signatures*, a term that already appeared in the literature (see e.g. [SK02])—however without providing a rigorous definition nor security proofs. As it is natural for proxy signatures, we consider a dynamic setting allowing to define non-frameability which we extend to additionally protect against wrongful accusation of delegation.

The most prominent example of a proxy signature scheme is "delegation-by-certificate": The delegator signs a document called the *warrant* containing the public key of the proxy and passes it to the latter. A proxy signature then consists of a regular signature by the proxy on the message and the signed warrant which together can by verified using the delegator's verification key only. Although not adaptable to the anonymous case—after all, the warrant contains the proxy's public key—, a virtue of the scheme is the fact that the delegator can restrict the delegated rights to specific *tasks* specified in the warrant. Since our model supports re-delegation, it is conceivable that a user wishes to re-delegate only a reduced subset of tasks she has been delegated for. We represent tasks by natural numbers and allow delegations for arbitrary sets of them, whereas re-delegation can be done for any subsets.

The primary practical motivation for the new primitive is GRID Computing, where Alice, after authenticating herself, starts a process. Once disconnected, the process may remain active, launch sub-processes and need additional resources that require further authentication. Alice thus delegates her rights to the process. On the one hand, not trusting the environment, she will not want to delegate all her rights, which can be realized by *delegation-by-certificate*. On the other hand, there is no need for the resources to know that it was not actually Alice who was authenticated, which is practically achieved solely by *full delegation*, i.e., giving the private key to the delegatee. While the first solution exposes the proxy's identity, the second approach does not allow for restriction of delegated rights

[1] Their scheme has later been attacked by [TL04]. Note, however, that our definition of non-frameability prevents this attack, since an adversary querying $\mathsf{PSig}(\cdot, \mathit{warr}, \cdot)$ and then creating a signature for task' is considered successful (cf. Sect. 3.3).

nor provide any means to trace malicious signers. Anonymous proxy signatures incorporate both requirements at one blow.

Another benefit of our primitive is that due to possible consecutiveness of delegations it can be regarded as *non-frameable, dynamic* hierarchical group signatures, a concept introduced by Trolin and Wikström [TW05] for the static setting.

After defining the new primitive and a corresponding security model, in order to show satisfiability of the definitions, we give an instantiation and prove it secure under the (standard) assumption that families of trapdoor permutations exist. The problem of devising a more efficient construction is left for future work. We emphasize furthermore that delegation in our scheme is non-interactive (the delegator simply sends a warrant she computed w.r.t. the delegatee's public key) and does not require a secure channel.

2 Algorithm Specification

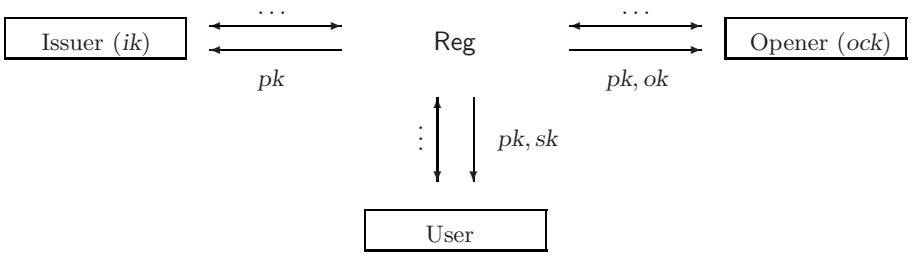
We describe an anonymous proxy signature scheme by giving the algorithms it consists of. First of all, running algorithm Setup with the security parameter λ creates the public parameters of the scheme, as well as the *issuing key ik* given to the issuer in order to register users and the opener's certification key *ock* given to potential openers. When a user registers, she and her opening authority run the interactive protocol Reg with the issuer. In the end, all parties hold the user's public key *pk*, the user is the only one to know the corresponding signing key *sk*, and the opener possesses *ok*, the key to open signatures on the user's behalf.

Once a user U_1 is registered and holds her secret key sk_1, she can delegate her signing rights to user U_2 holding pk_2 for a set of tasks *TList* by running $\mathsf{Del}(sk_1, \mathit{TList}, pk_2)$ to produce a warrant $warr_{1\to 2}$ enabling U_2 to proxy sign on behalf of U_1. Now if U_2 wishes to re-delegate the received signing rights for a possibly reduced set of tasks $\mathit{TList}' \subseteq \mathit{TList}$ to user U_3 holding pk_3, she runs $\mathsf{Del}(sk_2, warr_{1\to 2}, \mathit{TList}', pk_3)$, that is, with her warrant as additional argument, to produce $warr_{1\to 2\to 3}$. Every user in possession of a warrant valid for a task *task* can produce proxy signatures σ for messages M corresponding to *task* via $\mathsf{PSig}(sk, warr, task, M)$.[2] Anyone can then verify σ under the public key pk_1 of the first delegator (sometimes called "original signer" in the literature) by running $\mathsf{PVer}(pk_1, task, M, \sigma)$.

Finally, using the opening key ok_1 corresponding to pk_1, a signature σ can be opened via $\mathsf{Open}(ok_1, task, M, \sigma)$, which returns the list of users that have re-delegated as well as the proxy signer.[3] Note that for simplicity, we identify users with their public keys. Figure 1 gives an overview of the algorithms constituting an anonymous proxy signature scheme.

[2] Note that it depends on the concrete application to check whether M lies within the scope of *task*.

[3] We include *task* and M in the parameters of Open so the opener can verify the signature before opening it.



$$
\begin{array}{rcl}
\lambda & \rightarrow \mathsf{Setup} \rightarrow & pp, ik, ock \\
sk_x, [warr_{\rightarrow x},]\ TList, pk_y & \rightarrow \mathsf{Del} \rightarrow & warr_{[\rightarrow]x \rightarrow y} \\
sk_y, warr_{x \rightarrow \ldots \rightarrow y}, task, M & \rightarrow \mathsf{PSig} \rightarrow & \sigma \\
pk_x, task, M, \sigma & \rightarrow \mathsf{PVer} \rightarrow & b \in \{0,1\} \\
ok_x, \sigma, task, M \text{ and } \textit{registry}\text{-data} & \rightarrow \mathsf{Open} \rightarrow & \text{a list of users or } \perp \text{ (failure)}
\end{array}
$$

Fig. 1. Inputs and outputs of the algorithms

Consider a warrant established by executions of Del with correctly registered keys. Then for any task and message we require that the signature produced with it pass verification.

Remark (Differences to the Model for Proxy Signatures). The specification deviates from the one in [BPW03] in the following points: First, dealing with anonymous proxy signatures there is no general *proxy identification* algorithm; instead, only authorized openers holding a special key may revoke anonymity. Second, in contrast to the above specifications, the *proxy-designation protocol* in [BPW03] is a pair of interactive algorithms and the *proxy signing* algorithm takes a single input, the *proxy signing key skp*. However, by simply defining the proxy part of the proxy-designation protocol as

$$skp := (sk, warr)$$

any scheme satisfying our specifications is easily adapted to theirs.

3 Security Definitions

3.1 Anonymity

Anonymity ensures that signatures do not leak information on the identities of the intermediate delegators and the proxy signer. While this holds even in the presence of a corrupt issuer, the *number* of delegators involved may not remain hidden.

$\mathbf{Exp}^{\text{anon-b}}_{\mathcal{PS},A}(\lambda)$
$(pp, ik, ock) \leftarrow \mathsf{Setup}(1^\lambda)$
$(\text{ST}, pk, (sk^0, warr^0), (sk^1, warr^1), task, M)$
$\leftarrow A_1(pp, ik : \mathsf{USndToO}, \mathsf{ISndToO}, \mathsf{OK}, \mathsf{Open})$
if $pk \notin OReg$, return 0
for $c = 0 \,..\, 1$
$\sigma^c \leftarrow \mathsf{PSig}(sk^c, warr^c, task, M)$
if $\mathsf{PVer}(pk, task, M, \sigma^c) = 0$, return 0
$(pk_2^c, \ldots, pk_{k_c}^c) \leftarrow \mathsf{Open}(\mathsf{OK}(pk), task, M, \sigma^c)$
if opening succeeded and $k_0 \neq k_1$, return 0
$d \leftarrow A_2(\text{ST}, \sigma^b : \mathsf{Open})$
if A_1 did not query $\mathsf{OK}(pk)$ and A_2 did not query $\mathsf{Open}(pk, task, M, \sigma^b)$, return d, else return 0

Fig. 2. Experiment for ANONYMITY

A quite "holistic" approach to define anonymity is the following experiment in the spirit of CCA2-indistinguishability: The adversary A, who may control the issuer and all users, is provided with an oracle to communicate with an opening authority, who is assumed to be honest. A may also query opening keys and the opening of signatures. Eventually, he outputs a public key, a message, a task and two secret key/warrant pairs under one of which he is given a signature. Now A must decide which pair has been used to sign. Note that our definition implies all conceivable anonymity notions, such as proxy-signer anonymity, last-delegator anonymity, etc.

Figure 2 depicts the experiment, which might look more complex than expected, as there are several checks necessary to prevent the adversary from trivially winning the game by either

1. returning a public key he did not register with the opener,
2. returning an invalid warrant, that is, signatures created with it fail verification, or
3. having different lengths of delegation chains.[4]

The experiment simulates an honest opener as specified by Reg with whom the adversary communicates via the $\mathsf{USndToO}$ and $\mathsf{ISndToO}$ oracles, depending on whether he impersonates a user or the issuer. It also keeps a list $OReg$ of the opening keys created and the corresponding public keys. Oracle OK, called with a public key, returns the corresponding opening key from $OReg$ and when Open is called on $(pk', task', M', \sigma')$, the experiment looks up the corresponding

[4] The experiment checks 2. and 3. by using each of the returned warrants to create a signature, open both and check if the number of delegators match. Note, that traceability (cf. Sect. 3.2) guarantees that valid signatures can be opened.

opening key ok' and returns $\mathsf{Open}(ok', M', task', \sigma')$ if pk' has been registered and $\perp$ otherwise.

Definition 1 (Anonymity). *A proxy signature scheme $\mathcal{PS}$ is* ANONYMOUS *if for any probabilistic polynomial-time (p.p.t.) adversary $A = (A_1, A_2)$, we have*

$$\left| \Pr\left[\mathbf{Exp}^{\text{anon-1}}_{\mathcal{PS},A}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{anon-0}}_{\mathcal{PS},A}(\lambda) = 1\right] \right| \;=\; \text{negl}(\lambda) \;.$$

Remark (Hiding the Number of Delegations). A feature of our scheme is that users are able to delegate themselves. It is because of this fact—useful per se to create temporary keys for oneself for use in hostile environments—that one could define the following variant of the scheme:

Suppose there is a maximum number of possible delegations and that before signing, the proxy extends the actual delegation chain in her warrant to this maximum by consecutive self-delegations. The scheme would then satisfy a stronger notion of anonymity where even the number of delegations remains hidden. What is more, defining standard (non-proxy) signatures as self-delegated proxy signatures, even proxy and standard signatures become indistinguishable.

Since we also aim at constructing a generalization of group signatures in accordance with [BSZ05], we split the definition of what is called *security* in [BPW03] into two parts: traceability and non-frameability. We thereby achieve stronger security guarantees against malicious issuers.

3.2 Traceability

Consider a coalition of corrupt users and openers (the latter however following the protocol) trying to forge signatures. Then traceability guarantees that whenever a signature passes verification it can be opened.[5]

In the game for traceability we let the adversary A register corrupt users and see the communication between issuer and opener. To win the game, A must output a signature and a public key under which it is valid such that opening of the signature fails.

Figure 3 shows the experiment for traceability, where the oracles SndToI and SndToO simulate issuer and opener respectively, according to the protocol Reg. In addition, they return a transcript of the communication between them. The experiment maintains a list of generated opening keys, so OK returns the opening key associated to the public key it is called with, or $\perp$ in case the key is not registered—in which case Open returns $\perp$, too.

Definition 2 (Traceability). *A proxy signature scheme $\mathcal{PS}$ is* TRACEABLE *if for any p.p.t. adversary A, we have*

$$\Pr\left[\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda) = 1\right] \;=\; \text{negl}(\lambda) \;.$$

[5] The issuer is assumed to behave honestly as he can easily create unopenable signatures by registering dummy users and sign in their name. The openers are *partially* corrupt, otherwise they could simply refuse to open or not correctly register the opening keys.

$\mathbf{Exp}^{\text{trace}}_{\mathcal{PS},A}(\lambda)$
$(pp, ik, ock) \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk, task, M, \sigma) \leftarrow A(pp : \mathsf{SndToI}, \mathsf{SndToO})$
if $\mathsf{PVer}(pk, task, M, \sigma) = 1$ and $\mathsf{Open}(\mathsf{OK}(pk), task, M, \sigma) = \perp$
return 1, else return 0

Fig. 3. Experiment for TRACEABILITY

3.3 Non-frameability

Non-frameability ensures that no user is wrongfully accused of delegating or signing. In order to give a strong definition of non-frameability where we accord the adversary as much liberty as possible in his oracle queries, we require an additional functionality of the proxy signature scheme: Function OpenW applied to a warrant returns the list of delegators involved in creating it.

In the non-frameability game, the adversary can impersonate the issuer and the opener as well as corrupt users. He is given *all* keys created in the setup, and oracles to register honest users and query delegations and proxy signatures from them. To win the game, the adversary must output a task, a message and a valid signature on it, such that the opening reveals either

1. a second delegator or proxy signer who was never delegated by an honest original delegator for the task,
2. an honest delegator who was not queried the respective delegation for the task, or
3. an honest proxy signer who did not sign the message for the task and the respective delegation chain.

We emphasize that querying re-delegation from user U_2 to U_3 with a warrant from U_1 for U_2 and then producing a signature that opens to (U_1', U_2, U_3) is considered a success. Note furthermore that it is the adversary that chooses the opening key to be used. See Fig. 4 for the experiment for non-frameability.

ORACLES FOR NON-FRAMEABILITY: $\mathsf{ISndToU}$ ($\mathsf{OSndToU}$) enables the adversary impersonating a corrupt issuer (opener) to communicate with an honest user. When first called without arguments, the oracle simulates a user starting the registration procedure and makes a new entry in HU, the list of honest users. Oracles Del and PSig are called with a user's public key, which the experiment replaces by the user's secret key from HU before executing the respective function; e.g., calling Del with parameters $(pk_1, TList, pk_2)$ returns $\mathsf{Del}(sk_1, TList, pk_2)$. Oracle SK takes a public key pk as argument and returns the corresponding private key after deleting pk from HU.

Definition 3 (Non-frameability). *A proxy signature scheme $\mathcal{PS}$ is* NON-FRAMEABLE *if for any p.p.t. adversary A we have*

$\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda)$

$(pp, ik, ock) \leftarrow \mathsf{Setup}(1^\lambda)$

$(ok, pk_1, task, M, \sigma) \leftarrow A(pp, ik, ock : \mathsf{ISndToU}, \mathsf{OSndToU}, \mathsf{SK}, \mathsf{Del}, \mathsf{PSig})$

if $\mathsf{PVer}(pk_1, task, M, \sigma) = 0$ or $\mathsf{Open}(ok, task, M, \sigma) = \perp$, return 0

$(pk_2, \ldots, pk_k) = \mathsf{Open}(ok, task, M, \sigma)$

if $pk_1 \in HU$ and no queries $\mathsf{Del}(pk_1, TList, pk_2)$ with $TList \ni task$ made
return 1 (Case 1)

if for some $i \geq 2$, $pk_i \in HU$ and no queries $\mathsf{Del}(pk_i, warr, TList, pk_{i+1})$ with
$TList \ni task$ and $\mathsf{OpenW}(warr) = (pk_1, \ldots, pk_i)$ made, return 1 (Case 2)

if $pk_k \in HU$ and no queries $\mathsf{PSig}(pk_k, warr, task, M)$ made
with $\mathsf{OpenW}(warr) = (pk_1, \ldots, pk_{k-1})$ made, return 1 (Case 3)

return 0

Fig. 4. Experiment for Non-Frameability

$$\Pr\left[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1\right] \;=\; \text{negl}(\lambda) \;.$$

Remark. In the experiment $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$, the opening algorithm is run by the experiment, which by definition behaves honestly. To guard against a corrupt opener, it suffices to add a (possibly interactive) zero-knowledge proof to the system and have the opener prove correctness of opening.

4 An Instantiation of the Scheme

4.1 Building Blocks

To construct the generic scheme $\mathcal{PS}$, we will use the following cryptographic primitives (cf. Appendix A for the formal definitions) whose existence is implied by assuming trapdoor permutations [Rom90, DDN00, Sah99].

- $\mathcal{DS} = (\mathsf{K}_\sigma, \mathsf{Sig}, \mathsf{Ver})$, a digital signature scheme secure against existential forgeries under chosen-message attack [GMR88].
- $\mathcal{PKE} = (\mathsf{K}_\varepsilon, \mathsf{Enc}, \mathsf{Dec})$, a public-key encryption scheme with indistinguishable encryptions under adaptive chosen-ciphertext attack (CCA2) [RS92].
- $\Pi = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$, a non-interactive zero-knowledge (NIZK) proof system for an NP-language to be defined in the following that is simulation sound [BDMP91, Sah99].

4.2 Algorithms

The algorithm Setup establishes the public parameters and outputs the issuer's and the opener's certification key. The public parameters consist of the security parameter, a common random string for non-interactive zero-knowledge proofs

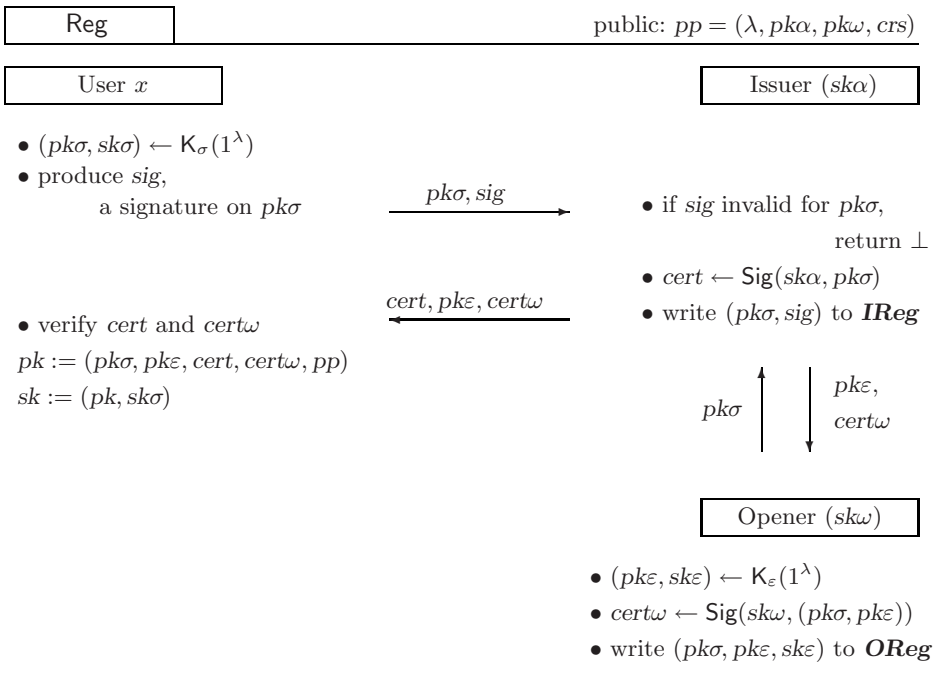


Fig. 5. Registration protocol

and the two signature verification keys corresponding to the issuer's and the opener's key:

Setup	
$1^\lambda \rightarrow$	$(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda);\ (pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda);\ crs \leftarrow \{0,1\}^{p(\lambda)}$
$pp, ik, ock \leftarrow$	$pp := (\lambda, pk\alpha, pk\omega, crs);\ ik := sk\alpha;\ ock := sk\omega$

The registration protocol is depicted in Fig. 5: When a user joins the system, she creates a pair of verification/signing keys $(pk\sigma, sk\sigma)$ and *signs* $pk\sigma$ (possibly via an external PKI) in order to commit to it. She then sends $pk\sigma$ and the signature sig to the issuer. The latter, after checking sig, signs $pk\sigma$ with his *certificate issuing key* $sk\alpha$ and writes the user data to ***IReg***, the registration table.

In addition, the issuer sends $pk\sigma$ to the authority responsible for opening the user's signatures. The opener creates an encryption/decryption key pair $(pk\varepsilon, sk\varepsilon)$ and a certificate on $pk\varepsilon$ and $pk\sigma$, which he sends together with $pk\varepsilon$ to the issuer, who forwards it to the user.[6]

[6] In practice, our protocol would allow for the opener to communicate directly with the user without the detour via the issuer—consider for example the case where each user is his own opener. We define the protocol this way to simplify exposition of the security proofs.

It is by having users create their own signing keys $sk\sigma$ that a corrupt authority is prevented from framing users. The user is however required to commit to her verification key via *sig*, so that she cannot later repudiate signatures signed with the corresponding signing key. Now to frame a user by creating a public key and attributing it to her, the issuer would have to forge *sig*. Note that it is impossible to achieve non-frameability without assuming some sort of PKI prior to the scheme.

Algorithm Del enables user x to pass her signing rights to user y (if called with no optional argument $\boldsymbol{warr}_{\text{old}}$), or to re-delegate the rights represented in $\boldsymbol{warr}_{\text{old}}$ for the tasks in *TList*. A warrant is an array where $\boldsymbol{warr}[i]$ corresponds to the i^{th} delegation and $\boldsymbol{warr}[i][task]$ contains basically a signature by the i^{th} delegator on the next delegator's public key and *task*.

More specifically, consider user x being the k^{th} delegator. If $k > 1$, she first copies all entries for the tasks to re-delegate from $\boldsymbol{warr}_{\text{old}}$ to the new warrant $\boldsymbol{warr}$. She then writes her public key to $\boldsymbol{warr}[k][0]$ that will later be used by the delegatee, and finally produces a signature on the task, the public keys of the delegators, her and the delegatee's public key and writes it to $\boldsymbol{warr}[k][task]$.

Del	
sk_x, $[\boldsymbol{warr}_{\text{old}}]$	parse $sk_x \rightsquigarrow (pk_x, sk\sigma)$; $k := \lvert\boldsymbol{warr}_{\text{old}}\rvert + 1$ // $k = 1$ if no $\boldsymbol{warr}_{\text{old}}$
$TList, pk_y \rightarrow$	for all $1 \le i < k$
	$\quad\boldsymbol{warr}[i][0] := \boldsymbol{warr}_{\text{old}}[i][0]$
	$\quad$for all $task \in TList$, $\boldsymbol{warr}[i][task] := \boldsymbol{warr}_{\text{old}}[i][task]$
	$\boldsymbol{warr}[k][0] := pk_x$
	for all $1 \le i \le k$, parse $\boldsymbol{warr}[i][0] \rightsquigarrow (pk\sigma_i, pk\varepsilon_i, cert_i, cert\omega_i, pp)$
	for all $task \in TList$
$\boldsymbol{warr} \leftarrow$	$\quad\boldsymbol{warr}[k][task] \leftarrow \mathsf{Sig}\big(sk\sigma, (task, pk\sigma_1, \ldots, pk\sigma_k, pk\sigma_y)\big)$

For every k, we define a relation R_k specifying an NP-language L_{R_k}. Basically, a theorem $(pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C)$ is in L_{R_k} if and only if

(1) $pk\varepsilon_1$ is correctly certified w.r.t. $pk\omega$,
(2) there exist verification keys $pk\sigma_2, \ldots, pk\sigma_k$ that are correctly certified w.r.t. $pk\alpha$,
(3) there exist warrant entries $warr_i$ for $1 \le i < k$, s.t. $pk\sigma_i$ verifies the delegation chain $pk_1 \rightarrow \cdots \rightarrow pk_{i+1}$,
(4) there exists a signature s on the delegation chain and M valid under $pk\sigma_k$,
(5) C is an encryption using some randomness ρ of all the verification keys, certificates, warrants and the signature s.

We define formally:

$$R_k\big[(pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C),\\ (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s, \rho)\big]$$

$$:\Leftrightarrow \quad \mathsf{Ver}\big(pk\omega, (pk\sigma_1, pk\varepsilon_1), cert\omega_1\big) = 1 \ \wedge \tag{1}$$

$$\textstyle\bigwedge_{2 \le i \le k} \mathsf{Ver}\big(pk\alpha, pk\sigma_i, cert_i\big) = 1 \ \wedge \tag{2}$$

$$\bigwedge_{1\le i\le k-1} \mathsf{Ver}\big(pk\sigma_i, (task, pk\sigma_1, \ldots, pk\sigma_{i+1}), warr_i\big) = 1 \wedge \quad (3)$$

$$\mathsf{Ver}\big(pk\sigma_k, (task, pk\sigma_1, \ldots, pk\sigma_k, M), s\big) = 1 \quad \wedge \quad (4)$$

$$\mathsf{Enc}\big(pk\varepsilon_1, (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s), \rho\big) = C \quad (5)$$

Note that for every k, the above relation R_k defines in fact an NP-language L_{R_k}, since given a witness, membership of a candidate theorem is efficiently verifiable and furthermore the length of a witness is polynomial in the length of the theorem. Let $\Pi_k := (\mathsf{P}_k, \mathsf{V}_k, \mathsf{Sim}_k)$ be a simulation-sound NIZK proof system for L_{R_k}.

Now to produce a proxy signature, it suffices to sign the delegation chain and the message, encrypt it together with all the signatures for the respective task from the warrant and prove that everything was done correctly, that is, prove that R_k is satisfied:

PSig

$sk, \boldsymbol{warr},$
$task, M \rightarrow$

$k := |\boldsymbol{warr}| + 1$, parse $sk \rightsquigarrow (pk_k, sk\sigma)$
parse $pk_k \rightsquigarrow \big(pk\sigma_k, pk\varepsilon_k, cert_k, cert\omega_k, (\lambda, pk\alpha, pk\omega, crs)\big)$
for $1 \le i < k$: parse $pk_i := \boldsymbol{warr}[i][0] \rightsquigarrow (pk\sigma_i, pk\varepsilon_i, cert_i, cert\omega_i, pp)$
set $warr_i := \boldsymbol{warr}[i][task]$
$s \leftarrow \mathsf{Sig}\big(sk\sigma, (task, pk\sigma_1, \ldots, pk\sigma_k, M)\big)$; $\rho \leftarrow \{0,1\}^{p_\varepsilon(\lambda,k)}$
$W := (pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s)$
$C \leftarrow \mathsf{Enc}(pk\varepsilon_x, W; \rho)$
$\pi \leftarrow \mathsf{P}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, warr\omega_1, task, M, C), W \,\|\, \rho, crs\big)$
$\sigma \leftarrow$ $\sigma := (C, \pi)$

Verifying a proxy signature then amounts to verifying the proof it contains:

PVer

$pk_x, task,$
$M, \sigma \rightarrow$

parse $pk_x \rightsquigarrow \big(pk\sigma_x, pk\varepsilon_x, cert_x, cert\omega_x, (\lambda, pk\alpha, pk\omega, crs)\big)$
$\sigma \rightsquigarrow (C, \pi)$
$b \leftarrow$ $b := \mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_x, pk\varepsilon_x, cert\omega_x, task, M, C),\ \pi,\ crs\big)$

To open a signature, after checking its validity, decrypt the ciphertext contained in it:

Open

$ok_x, task,$
$M, \sigma \rightarrow$

parse $ok_x \rightsquigarrow (pk_x, sk\varepsilon_x)$; $\sigma \rightsquigarrow (C, \pi)$
parse $pk_x \rightsquigarrow \big(pk\sigma_x, pk\varepsilon_x, cert_x, cert\omega_x, (\lambda, pk\alpha, pk\omega, crs)\big)$
if $\mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_x, pk\varepsilon_x, cert\omega_x, task, M, C),\ \pi,\ crs\big) = 0$
return $\perp$
$(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s)$
$:= \mathsf{Dec}(sk\varepsilon_x, C)$
$(pk_2, .., pk_k) \leftarrow$ if for some i, pk_i is not in $\boldsymbol{IReg}$, return $\perp$

4.3 Security Results

From the definition of the algorithms, it should be apparent that running PSig with a warrant correctly produced by registered users returns a signature which is accepted by PVer and correctly opened by Open. Moreover, the defined scheme satisfies all security notions from Sect. 3:

Lemma 1. *The proxy signature scheme $\mathcal{PS}$ is* ANONYMOUS *(Definition 1).*

Lemma 2. *The proxy signature scheme $\mathcal{PS}$ is* TRACEABLE *(Definition 2).*

Due to space limitations, we refer to the full version [FP08] for the proofs of Lemmata 1 and 2.

Lemma 3. *The proxy signature scheme $\mathcal{PS}$ is* NON-FRAMEABLE *(Definition 3).*

Proof (of Lemma 3).
Figure 6 shows experiment $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$ rewritten with the code of the respective algorithms. Note that we can dispense with the OSndToU-oracle, because in our scheme the user communicates exclusively with the issuer.

We construct an adversary B against the signature scheme $\mathcal{DS}$ having input a verification key $\overline{pk}$ and access to a signing oracle $\mathcal{O}_{\mathsf{Sig}}$. B simulates $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS}}$ for A, except that for one random user registered by A via ISndToU, B sets $pk\sigma$ to his input $\overline{pk}$, hoping that A will frame this very user. If B guesses correctly and A wins the game, a forgery under $\overline{pk}$ can be extracted from the proxy signature returned by A. Let $n(\lambda)$ be the maximal number of ISndToU queries A makes.

Adversary B and its handling of A's ISndToU and SK oracle queries or detailed in Fig. 6. To answer oracle calls Del and PSig with argument $pk^* = (\overline{pk}, \cdot\cdot)$, B replaces the line with $\mathsf{Sig}(sk\sigma, (task, pk\sigma_1, \ldots))$ in the respective algorithms by a query to his own signing oracle. For all other public keys, B holds the secret keys and can thus answer all queries.

Let S denote the event $\left[(pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C) \in L_R\right]$ and E_1, E_2, E_3 denote the union of S and the event that $\mathbf{Exp}^{\text{n-frame}}$ returns 1 in line 7, 8, 9, respectively. Then the following holds:[7]

$$\mathbf{Adv}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) \;\leq\; \Pr[E_1] \;+\; \Pr[E_2] \;+\; \Pr[E_3] \;+\; \Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1 \wedge \bar{S}]$$

We now show that the four summands are negligible:

1. Consider the event $E_1^* := [E_1 \wedge pk\sigma_1 = \overline{pk}]$. Then, since S is satisfied, we have $\mathsf{Ver}\big(\overline{pk}, (task, pk\sigma_1, pk\sigma_2), warr_1\big) = 1$,. So, B returns a valid message/signature pair.

   The forgery is valid, since B did not query its oracle for $(task, pk\sigma_1, pk\sigma_2)$ as this only happens when A queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_1, \cdot\cdot), \{\cdot\cdot, task, \cdot\cdot\}, (pk\sigma_2, \cdot\cdot))$, which by E_1 is not the case. Moreover, B simulates perfectly, for E_1 implies $\mathcal{O}_{\mathsf{SK}}((\overline{pk}, \cdot\cdot)$ was not queried. All in all, we have

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B} \;\geq\; \Pr[E_1^*] \;=\; \Pr[pk^* = pk_1] \cdot \Pr[E_1] \;=\; \tfrac{1}{n(\lambda)} \Pr[E_1]$$

[7] If not otherwise defined, we use $\mathbf{Adv}^{\bullet}_{\bullet\bullet}(\cdot)$ as shortcut for $\Pr[\mathbf{Exp}^{\bullet}_{\bullet\bullet}(\cdot) = 1]$.

$\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda)$

$(pk\alpha, sk\alpha) \leftarrow \mathsf{K}_\sigma(1^\lambda);\ (pk\omega, sk\omega) \leftarrow \mathsf{K}_\sigma(1^\lambda);\ crs \leftarrow \{0,1\}^{p(\lambda)}$
$pp := (\lambda, pk\alpha, pk\omega, crs)$
$(ok, pk, task, M, \sigma) \leftarrow A(pp, sk\alpha, sk\omega : \mathsf{ISndToU}, \mathsf{SK}, \mathsf{Del}, \mathsf{PSig})$
parse $ok \rightsquigarrow ((pk\sigma_1, pk\varepsilon_1, cert_1, cert\omega_1, pp), sk\varepsilon_1);\ \sigma \rightsquigarrow (C, \pi)$
if $\mathsf{V}_k\big(1^\lambda, (pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C), \pi, crs\big) = 0$ then return 0
$(pk\sigma_2, \ldots, pk\sigma_k, cert_2, \ldots, cert_k, warr_1, \ldots, warr_{k-1}, s) := \mathsf{Dec}(sk\varepsilon_1, C)$
if $pk_1 \in HU$ and no queries $\mathcal{O}_{\mathsf{Del}}(pk_1, \{\cdot\cdot, task, \cdot\cdot\}, pk_2)$ then return 1
if $\exists\, i : pk_i \in HU$ and no queries $\mathcal{O}_{\mathsf{Del}}(pk_i, warr, \{\cdot\cdot, task, \cdot\cdot\}, pk_{i+1})$
with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le i$ then return 1
if $pk_k \in HU$ and no queries $\mathcal{O}_{\mathsf{PSig}}(pk_k, warr, task, M)$
with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le k$ then return 1
return 0

$\mathcal{O}_{\mathsf{ISndToU}}(\emptyset)$

$(pk\sigma, sk\sigma) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
$HU := HU \cup \{(pk\sigma, sk\sigma)\}$
return $pk\sigma$

$\mathcal{O}_{\mathsf{SK}}((pk\sigma, \cdot\cdot))$

if $\exists\, sk\sigma : (pk\sigma, sk\sigma) \in HU$,
delete the entry and return $sk\sigma$
otherwise, return $\perp$

Adversary $B(\overline{pk} : \mathsf{Sig}(sk, \cdot))$

$j^* \leftarrow \{1, \ldots, n\};\ j := 0$
$\vdots$
if $pk\sigma_1 = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{Del}}((pk_1, \cdot\cdot), \{\cdot\cdot, task, \cdot\cdot\}, (pk\sigma_2, \cdot\cdot))$
then return $\big((task, pk\sigma_1, pk\sigma_2), warr_1\big)$
if $\exists\, i : pk\sigma_i = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_i, \cdot\cdot), warr, \{\cdot\cdot, task, \cdot\cdot\}, (pk\sigma_{i+1}, \cdot\cdot))$
with $warr[j][0][1] = pk\sigma_j$ for $1 \le j \le i$
then return $\big((task, pk\sigma_1, \ldots, pk\sigma_{i+1}), warr_i\big)$
if $pk\sigma_k = \overline{pk}$ and no queries $\mathcal{O}_{\mathsf{PSig}}((pk\sigma_k, \cdot\cdot), warr, task, M)$ with
$warr[j][0][1] = pk\sigma_j$ for $1 \le j \le k$, then return $\big((task, pk\sigma_1, \ldots, pk\sigma_k, M), s\big)$
return 0

$\mathcal{O}_{\mathsf{ISndToU}}(\emptyset)$ by B

$j := j + 1$; if $j = j^*$, return $\overline{pk}$
$(pk\sigma, sk\sigma) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
$HU := HU \cup \{(pk\sigma, sk\sigma)\}$
return $pk\sigma$

$\mathcal{O}_{\mathsf{SK}}((pk\sigma, \cdot\cdot))$ by B

if $pk\sigma = \overline{pk}$ then abort
else if $\exists\, sk\sigma : (pk\sigma, sk\sigma) \in HU$
delete entry, return $sk\sigma$
return $\perp$

Fig. 6. Instantiated experiment for non-frameability and adversary B against $\mathcal{DS}$

2. Consider the event $[E_2 \wedge pk\sigma_i = \overline{pk}]$: Then S implies
$$\mathsf{Ver}(\overline{pk}, \big((task, pk\sigma_1, \ldots, pk\sigma_{i+1}), warr_i\big) = 1$$
So, B returns a valid signature on a message he did not query its signing oracle: Only if A queries $\mathcal{O}_{\mathsf{Del}}((pk\sigma_i, \cdot\cdot), warr, \{\cdot\cdot, task, \cdot\cdot\}, (pk\sigma_{i+1}, \cdot\cdot))$ with $warr[j][0][1] = pk\sigma_j$ for $1 \leq j \leq i+1$, B queries $(task, pk\sigma_1, \ldots, pk\sigma_{i+1})$. Moreover, B simulates perfectly, as there was no query $\mathcal{O}_{\mathsf{SK}}((\overline{pk}, \cdot\cdot)$. As for 1., we have $\frac{1}{n(\lambda)} \Pr[E_2] \leq \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B}$.
3. Consider the event $[E_3 \wedge pk\sigma_k = \overline{pk}]$: There were no $\mathcal{O}_{\mathsf{SK}}((\overline{pk}, \cdot\cdot)$ queries and by S, B outputs a valid pair. B did not query $(task, pk\sigma_1, \ldots, pk\sigma_k, M)$ (as A made no query $\mathcal{O}_{\mathsf{PSig}}((pk\sigma_k, \cdot\cdot), warr, task, M)$ with $warr[j][0][1] = pk\sigma_j$ for $1 \leq j \leq k$). Again, we have $\frac{1}{n(\lambda)} \Pr[E_3] \leq \mathbf{Adv}^{\text{euf-cma}}_{\mathcal{DS},B}$
4. The event $\Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1]$ implies
$$\mathsf{V}_k(1^\lambda, (pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C), \pi, crs) = 1,$$
which, together with $\bar{S}$ contradicts soundness of Π: based on $\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}$, we could construct an adversary B_{s} against soundness of Π which after receiving crs (rather than choosing it itself), runs along the lines of the experiment until Line 4 and then outputs $\big((pk\alpha, pk\omega, pk\sigma_1, pk\varepsilon_1, cert\omega_1, task, M, C), \pi\big)$. We have thus
$$\Pr[\mathbf{Exp}^{\text{n-frame}}_{\mathcal{PS},A}(\lambda) = 1 \wedge \bar{S}] \;\leq\; \mathbf{Adv}^{\text{ss}}_{\Pi,B_{\text{s}}}$$ □

Theorem 1. *Assuming trapdoor permutations, there exists an anonymous traceable non-frameable proxy signature scheme.*

Proof. Follows from Lemmata 1, 2 and 3. □

We have thus defined a new primitive unifying the concepts of group and proxy signatures and given strong security definitions for it. Moreover, Theorem 1 shows that these definitions are in fact satisfiable in the standard model, albeit by a inefficient scheme. We are nonetheless confident that more practical instantiations of our model will be proposed, as it was the case for group signatures; see e.g. [BW07] for an efficient instantiation of a variation of the model by [BMW03]. We believe in particular that the novel methodology to construct NIZK proofs introduced by [GS08] will lead to practically usable implementations.

Acknowledgments

This work was partially funded by EADS, CELAR, ANR PAMPA and ECRYPT.

References

[BMW03] Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)

[BSZ05] Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005)

[BDMP91] Blum, M., De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge proof systems. SIAM Journal on Computing 20(6), 1084–1118 (1991)

[BPW03] Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. IACR ePrint Archive: Report 2003/096 (2003)

[BW07] Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)

[CvH91] Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

[DDN00] Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)

[FP08] Fuchsbauer, G., Pointcheval, D.: Anonymous Proxy Signatures, `http://www.di.ens.fr/homedirfuchsbau`

[GMR88] Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing 17(2), 281–308 (1988)

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[MUO96] Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS). ACM, New York (1996)

[RS92] Rackoff, C., Simon, D.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)

[RST01] Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)

[Rom90] Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: 22nd Annual Symposium on Theory of Computing, pp. 387–394. ACM, New York (1990)

[Sah99] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Symposium on Foundations of Computer Science, pp. 543–553. IEEE, Los Alamitos (1999)

[SK02] Shum, K., Wei, V.K.: A strong proxy signature scheme with proxy signer privacy protection. In: 11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2002), pp. 55–56. IEEE, Los Alamitos (2002)

[TL04] Tan, Z., Liu, Z.: Provably secure delegation-by-certification proxy signature schemes. IACR ePrint Archive: Report 2004/148 (2004)

[TW05] Trolin, M., Wikström, D.: Hierarchical group signatures. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 446–458. Springer, Heidelberg (2005)

A Formal Definitions of the Employed Primitives

A.1 Signature Scheme $\mathcal{DS} = (\mathsf{K}_\sigma, \mathsf{Sig}, \mathsf{Ver})$

$\mathcal{DS}$ is a digital signature scheme, that is

$$\forall\, \lambda \in \mathbb{N}\ \forall\, m \in \{0,1\}^*\ \forall\, (pk, sk) \leftarrow \mathsf{K}_\sigma(1^\lambda) :\ \mathsf{Ver}\big(pk, m, \mathsf{Sig}(sk, m)\big) = 1$$

We assume $\mathcal{DS}$ is secure against *existential forgery under chosen-message attack*, that is

$$\forall \text{ p.p.t. } A :\ \Pr\left[\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},A}(\lambda) = 1\right] \ =\ \text{negl}(\lambda) \qquad \text{with}$$

$\mathbf{Exp}^{\text{euf-cma}}_{\mathcal{DS},A}(\lambda)$
$(pk, sk) \leftarrow \mathsf{K}_\sigma(1^\lambda)$
$(m, \sigma) \leftarrow A(pk : \mathsf{Sig}(sk, \cdot))$
if $\mathsf{Ver}(pk, m, \sigma) = 1$ and A never queried m, return 1, else return 0

A.2 Public-Key Encryption Scheme $\mathcal{PKE} = (\mathsf{K}_\varepsilon, \mathsf{Enc}, \mathsf{Dec})$

$\mathcal{PKE}$ is a public-key encryption scheme, that is

$$\forall\, \lambda \in \mathbb{N}\ \forall\, m \in \{0,1\}^*\ \forall\, (pk, sk) \leftarrow \mathsf{K}_\varepsilon(1^\lambda) :\ \mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$$

We assume that $\mathcal{PKE}$ satisfies *indistinguishability under adaptive chosen-ciphertext attacks*, i.e.,

$\forall$ p.p.t. $A = (A_1, A_2)$:
$\left|\Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathcal{PKE},A}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathcal{PKE},A}(\lambda) = 1\right]\right| \ =\ \text{negl}(\lambda) \qquad \text{with}$

$\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{PKE},A}(\lambda)$
$(pk, sk) \leftarrow \mathsf{K}_\varepsilon(1^\lambda)$
$(m_0, m_1, \text{ST}) \leftarrow A_1(pk : \mathsf{Dec}(sk, \cdot))$
$y \leftarrow \mathsf{Enc}(pk, m_b)$
$d \leftarrow A_2(\text{ST}, y : \mathsf{Dec}(sk, \cdot))$
if $|m_0| = |m_1|$ and A_2 never queried y return d, else return 0

A.3 Non-interactive Zero-Knowledge Proof System $\Pi = (\mathsf{P}, \mathsf{V}, \mathsf{Sim})$ for L_R

We require that Π satisfy the following properties:

– Completeness $\quad \forall\, \lambda \in \mathbb{N}\ \ \forall\, (x, w) \in R$ with $|x| < \ell(\lambda)\ \ \forall\, r \in \{0,1\}^{p(\lambda)}$:

$$\mathsf{V}\big(1^\lambda, x, \mathsf{P}(1^\lambda, x, w, r), r\big) = 1$$

– Soundness $\forall$ p.p.t. A :

$$\Pr\left[r \leftarrow \{0,1\}^{p(\lambda)};\ (x,\pi) \leftarrow A(r) :\ x \notin L \wedge \mathsf{V}(1^\lambda, x, \pi, r) = 1\right] \ =\ \mathrm{negl}(\lambda)$$

– Adaptive Single-Theorem Zero Knowledge $\forall$ p.p.t. A :

$$\mathbf{Adv}^{\mathrm{zk}}_{\Pi,A}(\lambda) := \left|\Pr\left[\mathbf{Exp}^{\mathrm{zk}}_{\Pi,A}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{zk-S}}_{\Pi,A}(\lambda) = 1\right]\right| = \mathrm{negl}(\lambda) \quad \text{with}$$

$\mathbf{Exp}^{\mathrm{zk}}_{\Pi,A}(\lambda)$
$r \leftarrow \{0,1\}^{p(\lambda)}$
$(x, w, \mathrm{ST}_A) \leftarrow A_1(r)$
$\pi \leftarrow \mathsf{P}(x, w, r)$
return $A_2(\mathrm{ST}_A, \pi)$

$\mathbf{Exp}^{\text{zk-S}}_{\Pi,A}(\lambda)$
$(r, \mathrm{ST}_S) \leftarrow \mathsf{Sim}_1(1^\lambda)$
$(x, w, \mathrm{ST}_A) \leftarrow A_1(r)$
$\pi \leftarrow \mathsf{Sim}_2(\mathrm{ST}_S, x)$
return $A_2(\mathrm{ST}_A, \pi)$

– Simulation Soundness

$$\forall \text{ p.p.t. } A :\ \Pr\left[\mathbf{Exp}^{\mathrm{ss}}_{\Pi,A}(\lambda) = 1\right] \ =\ \mathrm{negl}(\lambda) \qquad \text{with}$$

$\mathbf{Exp}^{\mathrm{ss}}_{\Pi,A}(\lambda)$
$(r, \mathrm{ST}_S) \leftarrow \mathsf{Sim}_1(1^\lambda)$
$(y, \mathrm{ST}_A) \leftarrow A_1(r)$
$\pi \leftarrow \mathsf{Sim}_2(\mathrm{ST}_S, y)$
$(x, \pi') \leftarrow A_2(\mathrm{ST}_A, \pi)$
if $\pi \neq \pi'$ and $x \notin L_R$ and $\mathsf{V}(1^\lambda, x, \pi', r) = 1$ return 1, else return 0.

Multisignatures Using Proofs of Secret Key Possession, as Secure as the Diffie-Hellman Problem*

Ali Bagherzandi and Stanisław Jarecki

Department of Computer Science,
University of California, Irvine
{zandi,stasio}@ics.uci.edu

Abstract. A multisignature scheme allows a group of n players to produce a short string which is equivalent to n separate signatures on the same message. Assuming the Random Oracle Model (ROM), the aggregate signature schemes of Boneh et al. [BGLS03] and Bellare and Neven [BN06] provide multisignatures secure in the standard public key setting, but their multisignature verification algorithms involve respectively $O(n)$ bilinear maps and $O(n)$ exponentiations. Ristenpart and Yilek [RY07] recently showed two multisignature schemes relying on groups with bilinear maps, with just $O(1)$ bilinear maps in multisignature verification, which are secure if each public key is accompanied by so-called "proof of (secret key) possession" (POP). We show how to achieve secure multisignatures in the POP model using any group where CDH or DDH problems are hard. Both schemes have multisignature verification with $O(1)$ exponentiations, and their POP messages take $O(1)$ group elements and require $O(1)$ exponentiations to verify. Moreover, the security of the proposed schemes is *tightly* related to the CDH and DDH problems, in ROM.

1 Introduction

A multisignature scheme allows a group of n players to sign a common message so that instead of n separate signatures the players produce a short string which can be verified against the set of the public keys of the participating players. Such scheme is interesting if the resulting string is shorter than n separate signatures and/or the verification time is faster than n separate signature verifications. Applications of multisignatures include scenarios where the number of signers is moderate, like co-signing, distribution of certificate authorities, or aggregation of PKI certificate chains. However, multisignatures can potentially be useful also in very large groups of signers, *e.g.* for aggregation of acknowledgements in response to a broadcast.

Rogue Key Attacks and the KOSK Assumption. Multisignature schemes are possible because of homomorphic properties of arithmetic operations involved in standard signatures. For example, a BLS signature [BLS04] on message m under public key $y_i = g^{x_i}$ is $\sigma_i = H(m)^{x_i}$, i.e. a value s.t. $(g, y_i, H(m), \sigma_i)$ is a DDH tuple. A corresponding multisignature can be created as $\sigma = \prod_{i=1}^{n} \sigma_i$, and it can be verified under the

* Research supported by NSF CyberTrust Grant #0430622.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 218–235, 2008.

combined public key $y = \prod_{i=1}^{n} y_i$, because $(g, y, H(m), \sigma)$ is also a DDH tuple. Unfortunately, the same homomorphic properties which enable aggregation of signatures into multisignatures can enable a "rouge key attack" on such schemes. For example, the above scheme is insecure because an adversary who picks $y_2 = g^x/y_1$ for some existing key y_1 and any x can use $x = DL(g, y_1 * y_2)$ to issue multisignatures on behalf of key set $\{y_1, y_2\}$. Indeed, as Micali et al. [MOR01] point out, many proposed multisignature schemes are vulnerable to such rouge key attacks, *e.g.* [LHL94, Har94], or their security requires trusted generation of each key, *e.g.* [OO91, OO99]. Interestingly, rouge key attackers usually do not know the private key corresponding to the rogue public key. Indeed, under the discrete logarithm assumption it is provably hard to compute the private key x_2 *s.t.* $y_2 = g^{x_2}$ in the above attack. This observation led Micali, Ohta, and Reyzin [MOR01] to construct the first multisignature scheme secure without assuming trusted key generation. However, that scheme requires all potential signers to engage in an interactive initialization protocol in which every player proves knowledge of its secret key to all others, and such initialization procedure does not tolerate dynamic groups and does not scale well to large groups. One way to remove this initialization procedure is to assume the so-called *knowledge of secret key* (KOSK) assumption [Bol03] on key registration process: The KOSK assumption states that if an adversary registers a public key then the adversary's algorithm must also explicitly output a corresponding secret key. Two secure multisignature schemes were proposed under this assumption using bilinear maps, by Boldyreva [Bol03] in ROM and by Lu et al. [LOS$^+$06] in the standard model (*i.e.* without ROM).

Multisignatures in the Key Registration Model. One way to realize the KOSK assumption is to employ so-called *Key Registration Model* (KR) for Public Key Infrastructure (PKI), introduced in the context of multisignatures by Ristenpart and Yilek [RY07]. In the KR model for PKI, a CA issues a certificate on a key only if its owner passes a special key registration protocol. For example, the PKCS#10 [PKC00] standard for CA operation asks the user to sign a challenge message under its public key. This challenge-signature pair is called a *proof of possession* of the secret key (POP) in PKCS#10, but we'll use this term more broadly, for any user-generated string verified by either the CA or by multisignature verifiers (see the "Key Verification" model below). The intuitive goal of the POP mechanism in PKCS#10 was to assure that someone has access to the secret key corresponding to the public key being certified, but this mechanism does not implement the KOSK assumption in general. Indeed, Ristenpart and Yilek [RY07] showed that the schemes of [Bol03, LOS$^+$06] are insecure in the KR model if key registration is implemented with POPs of PKCS#10. Nevertheless, [RY07] also showed that using a slight variant of the same POP mechanism the schemes of [Bol03, LOS$^+$06] yield secure multisignature schemes in the KR model, relying on bilinear maps.

Alternatively, one can realize the KOSK model by implementing POPs with concurrently secure zero-knowledge proofs of knowledge (ZKPK) of a secret key. Such ZKPK's can be achieved in ROM by the results of Fischlin [Fis05] using $O(\log \kappa)$ group elements where κ is the security parameter. Combined with the multisignature protocol of [MOR01], this implies a multisignature scheme in the KR model secure under the DL assumption. However, non-constant-sized POPs are less practical if POP messages are verified by multisignature receivers instead of by the CA's (see the "Key

Registration vs. Key Verification" below). Moreover, due to the heavy use of the forking lemma in the reduction of [MOR01], the exact security of this scheme is not optimal.

Multisignatures in the Plain Public Key Model. One of the drawbacks of multisignatures in the KR model is that they require modifications to the current operation of the CA's. (In addition to imposing non-standard trust requirements on CA's, as we argue below.) It is therefore highly interesting to provide multisignature schemes which are secure in the *plain* setting where no special registration process is assumed for public keys. The first such scheme is implied in ROM by an aggregate signature scheme of Boneh et al. [BGLS03], with the analysis extended by Bellare et al. [BNN07]), but its multisignature verification algorithm requires $O(n)$ bilinear map operations. Bellare and Neven recently proposed a multisignature secure in the plain setting which does bilinear maps [BN06]. While this scheme has a significantly smaller cost of multisignature verification, it still requires $O(n)$ exponentiations: The way [BN06] avoid KOSK and KR models is by using independent challenges in the proofs of knowledge of discrete logarithm performed by each player in the multisignature generation protocol. However, the multisignature verification operation then becomes a multi-exponentiation on n public keys and n different exponents, and to the best of our knowledge the cost of such multiexponentiation is still $O(n)$ the cost of a single exponentiation.

Key Registration vs. Key Verification. The fact that current multisignatures in the plain setting have slower verification than current schemes secure in the KR model motivates looking closer at the KR model. For example, to the best of our knowledge it has not been previously observed that the Key Registration model requires non-standard trust assumptions among the PKI participants. Consider an application of a multisignature scheme, where a multisignature is formed by some users certified by CA's trusted by the multisignature verifier, and some certified by CA's who are unknown to this verifier. Assume that the verifier is interested in checking whether or not the message was signed by all the users of the first type but does not care if users of the second type have also contributed to the multisignature. An example is a petition signed by individuals whose public keys are certified by different CA's, some widely known and trusted, some less so. If a multisignature scheme secure in the KR model is used then the verifier cannot conclude that the message was signed by the users she cares about, certified by the CA's she recognizes and trusts as long as a single participant in the multisignature is certified by a CA which she does not recognize and/or trust. This is because the scheme provides no security guarantees if the prescribed key registration procedure, e.g. POP verification, is not followed with regards to even a single key participating in the multisignature. This imposes a limitation on the use of multisignatures compared to standard signatures or multisignatures secure in the plain setting, since in either of the last two cases the verifier can decide if the users she cares about signed the petition whether or not it was also signed by keys certified by unknown or suspect CA's.

We propose to remove this limitation in the usage of multisignatures secure in the KR model by considering an alternative mode of PKI operation which we call the *Key Verification* (KV) Model. In the KV model each private key owner also produces a POP string, but instead of handing it to the CA during the key registration process she attaches it to her key (or a PKI certificate on the key). This POP message is then verified by a multisignature receiver instead of by the CA, for example together with verification

of PKI certificates on that key. We note that in the multisignature schemes we propose POP verification costs are comparable to DSA certificate verification, and this cost can be further reduced by batching. The Key Verification model of operation should also make it easier to adopt multisignature schemes: Since the CA operation does not need to change, a multisignature scheme secure in the KV model can potentially use existing private-public keys and certificates. For example, our CDH-based multisignature scheme can utilize existing DSA or Schnorr signature public keys. We stress that while the KR and KV models differ in how a multisignature scheme operates, any multisignature scheme secure in the KV model trivially implies a scheme secure in the KR model, and any scheme secure in the KR model which has a non-interactive key registration process (e.g. the schemes given by [RY07]) implies a scheme secure in the KV model.

Our Contributions. We propose two multisignature schemes in the KV (or KR) model, denoted MDDH and MCDH. Both schemes take three rounds, have multisignature verification procedures with $O(1)$ exponentiations, do not require bilinear maps, and their security is *tightly* related in ROM to, respectively, the CDH and DDH problems. The POP messages in both schemes take $O(1)$ group elements and their verification takes $O(1)$ exponentiations. Figure 1 summarizes the comparison between ours and previous multisignature schemes. In this table, RY+BLS and RY+Waters refers to the first and the second schemes of [RY07] respectively, BGLS refers to the multisignature scheme implied by the aggregate signature proposed by Boneh et al [BGLS03], MOR+Fischlin refers to the scheme of Micali et al [MOR01] with its initialization phase replaced by key registration using Fischlin's ZKPK's [Fis05], and BN refers to the scheme proposed by Bellare and Neven [BN06].

Compared to the two schemes of [RY07] our schemes do not rely on groups with bilinear maps, but they do so at the cost of relying on the ROM model, which one of the schemes of [RY07] avoids, and by using an interactive multisignature generation. This drawback is significant in many applications but it can be mitigated in applications where the same set of players is expected to repeatedly engage in several instances of the multisignature protocol, as long as the multisignature procedure is fast on-line, *i.e.* if the signed message is an input to the players only in the last round of the interaction, which is the case with our DDH-based scheme. (It is an open problem whether the CDH-based scheme can be made fast on-line without sacrificing other parameters.)

In comparison with the DL-based scheme in the Key Verification model implied by the combined results of [MOR01, Fis05], our POP messages are shorter and faster to verify, which is especially important in the Key Verification model where POP's must be attached to public keys and verified by multisignature recipients. To achieve 2^{80} security the POP size and verification time in the scheme implied by [MOR01, Fis05] would be larger by roughly a factor of ten when compared to our CDH-based and DDH-based schemes. Moreover, the security reduction from the DL problem implied by these two results is inexact, while our schemes have exact reductions from CDH or DDH problems, and in many groups of interest the DL and CDH problems are almost equivalent [MW00].

Finally, compared to the scheme of [BN06] which works in a plain model, our schemes require a Key Verification model. This is a drawback, but as discussed in a subsection above, in many scenarios the Key Verification model of PKI operation

MS Scheme	Assumption on Security(1)	Degradation in Security(2)	Protocol Rounds	Key Setup	Sig.Ver. Time(3)	Signing Time(3)	Signature Length(4)
RY+BLS	GapDH	$1/q_s$	1	POP	O(1)	O(1)	$\|G_1\|$
RY+Waters	GapDH	$1/q_s$	1	POP	O(1)	O(1)	$\|G_1\| + \|G_2\|$
BGLS	GapDH	$1/q_s$	1	Plain	O(n)	O(1)	$\|G_1\|$
MOR+Fischlin	DL(5)	$1/q_s{q_h}^2$	2	POP	O(1)	O(1)	$2\|q\|$
BN	DL(5)	$1/q_h$	3	Plain	O(n)	O(1)	$\|G\| + \|q\|$
MDDH	DDH	exact	3	POP	O(1)	O(1)	$2\|q\|$
MCDH	CDH(6)	exact	3	POP	O(1)	O(n)	$\|G\| + 2\|q\| + 2\kappa$

Fig. 1. (1) All schemes except RY+Waters assume a ROM model; (2) Security degradation is given as a factor f *s.t.* if a multisignature adversary succeeds with probability ϵ then the reduction breaks its underlying security assumption with probability $\Omega(f*\epsilon)$ in comparable time. Here q_s and q_h denote, respectively, the number of adversary's signature and hash queries; (3) Computational costs are the number of modular exponentiations (or bilinear maps for the GapDH-based schemes); (4) Signature length is measured in bits, where κ is the security parameter, $|G|$ is the number of bits required to represent elements in group G, q is the group order, and G_1 and G_2 are two groups of points on an elliptic curve with asymmetrical bilinear maps. For example $\kappa = 80$, $|G| = |q| = |G_1| = 160$ and $|G_2| = 6 * 160$; (5) The reduction for the Fischlin+MOR scheme is our best estimate. The reduction given in [BN06] for the BN scheme has ϵ/q_h degradation, but it seems that one can modify it to provide only $1/q_h$ degradation using the version of the forking lemma originally given by Pointcheval and Stern [PS00]; (6) For the MCDH protocol we only give an exact security reduction from the *expected-time* hardness of the CDH problem.

creates a small overhead in the certificate verification process. On the other hand, our schemes have tight reductions from CDH/DDH problems while the security reduction of [BN06] encounters a security degradation due to the use of the forking lemma, and our schemes make $O(1)$ exponentiation operations during multisignature verification compared to $O(n)$ exponentiation cost in [BN06]. We stress that while it might seem that our schemes require $O(n)$ verification, because we require that each multisignature verifier checks all n POP messages attached to the certificates of the n players involved in the multisignature, the $O(1)$ multisignature verification in the KV model is better than $O(n)$ verification in the plain model for two reasons: (1) Since every entity in PKI normally would keep a hash table of previously verified keys, the initial cost of key verification amortizes over all multisignature verifications which involve this key. (2) If the CA's use DL-based certificates, then the cost of key verification imposed by our schemes is only a constant factor higher than the cost of certificate verification.

In terms of multisignature size, our DDH-based scheme is the same as that of [BN06], and the multisignature in our CDH-based scheme is about 2 times larger than the multisignature of [BN06], when implemented over elliptic curves. Note, however, that if one takes the exact security results into account, the two schemes achieve the same level of provable security when the scheme of [BN06] is implemented over twice larger groups, assuming the near equivalence of the DL and CDH problems. Unlike all other multisignature schemes discussed, our CDH based scheme requires $O(n)$ signing time per party due to verification of NIZKs generated by each player. This may or may not be a drawback in different applications, but it seems that the communication costs of multisignature generation would often trump this computational cost. We point out that

our CDH based scheme has a tight reduction only from expected-time hardness of the CDH problem. However, in generic groups the expected-time hardness of the CDH is essentially the same as the fixed-time hardness of this problem, i.e. for every algorithm which solves the CDH problem in a generic group of size q with probability ϵ and expected-time T, it holds that $T/\epsilon \leq \sqrt{q}$.

Organization. After preliminaries in Section 2 we define secure multisignatures in the Key Verification model in Section 3, and present our DDH-based and CDH-based multisignature schemes respectively in Sections 4 and 5.

2 Preliminaries: Notation and Assumptions

Let G be a multiplicative group of a prime order q, and let g be its generator. All arithmetic operations are either modulo q, when involving numbers chosen in $\mathbb{Z}_q$, or they are operations in G, when involving group elements. We use notation $x \xleftarrow{r} S$ to denote a probabilistic process which assigns to variable x a uniformly chosen value in set S. We write $a|b$ to denote the concatenation of bit strings a and b.

The computational Diffie Hellman (CDH) problem in group G is a problem of computing g^{xy}, given the tuple (g, g^x, g^y) for random x, y in $\mathbb{Z}_q$, while the decisional Diffie Hellman (DDH) problem in G is the problem of distinguishing between tuples of the form (g, g^x, g^y, g^{xy}) for random x, y in $\mathbb{Z}_q$, and (g, g^x, g^y, g^z) for random x, y, z in $\mathbb{Z}_q$.

Definition 1. *The CDH problem is (t, ϵ)-hard in G if for any algorithm B running in time t, we have $\mathbf{Adv}_G^{\mathsf{CDH}}(\mathcal{B}) \leq \epsilon$ where:*

$$\mathbf{Adv}_G^{\mathsf{CDH}}(\mathcal{B}) = \Pr_{x,y \xleftarrow{r} \mathbb{Z}_q}[\mathcal{B}(g, g^x, g^y) = g^{xy}]$$

Definition 2. *The DDH problem is (t, ϵ)-hard in G if for any algorithm B running in time t, we have $\mathbf{Adv}_G^{\mathsf{DDH}}(\mathcal{B}) \leq \epsilon$ where:*

$$\mathbf{Adv}_G^{\mathsf{DDH}}(\mathcal{B}) = |\Pr_{x,y \xleftarrow{r} \mathbb{Z}_q}[\mathcal{B}(g, g^x, g^y, g^{xy}) = 1] - \Pr_{x,y,z \xleftarrow{r} \mathbb{Z}_q}[\mathcal{B}(g, g^x, g^y, g^z) = 1]|$$

Hardness of DL problem in G is implied by the hardness of either the DDH problem or the CDH problem in G, but the converse is not known to be true. However, all these problems have the same hardness as the DL problem in *generic groups* [Sho00]. Moreover, by the results of Maurer and Wolf [MW99], the CDH problem is very closely related to the DL problem in a large class of groups that are commonly used in cryptography. Also, see [Bon98] for various groups where DDH assumption might hold.

3 Multisignature Schemes

We define a Multisignature Scheme (MS) in the Key Verification (KV) model as a tuple $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{KVrfy}, \mathsf{MSign}, \mathsf{Vrfy})$ where Setup, KGen, KVrfy and Vrfy are efficient probabilistic algorithms and MSign is an interactive protocol *s.t.*

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$, on input the security parameter κ generates the public parameters par.
- $(sk, pk, \pi) \leftarrow \mathsf{KGen}(\mathsf{par})$, executed by each user on input par, generates this user's secret key sk, the corresponding public key pk, and a proof of validity of this public key, denoted π.
- $\{0,1\} \leftarrow \mathsf{KVrfy}(\mathsf{par}, pk, \pi)$ verifies whether pk is a valid key, given the proof π.
- MSign is a multisignature protocol executed by a group of players who intend to sign the same message m. Each player P_i executes MSign on public inputs par, message m and private input sk_i, its secret key, and outputs a multisignature σ.
- $\{0,1\} \leftarrow \mathsf{Vrfy}(\mathsf{par}, m, \mathsf{PKSet}, \sigma)$ verifies whether σ is a valid multisignature on message m on behalf of the set of players whose public keys are in set PKSet.

The above set of procedures must satisfy the following *correctness* property. Let par be any output of $\mathsf{Setup}(1^\kappa)$. First, any (sk, pk, π) output by $\mathsf{KGen}(\mathsf{par})$ satisfies $\mathsf{KVrfy}(\mathsf{par}, pk, \pi) = 1$. Second, for any $n \leq n_{max}$, any message m, and any (sk_i, pk_i, π_i) tuples, $i \in \{1, ..., n\}$, generated by $\mathsf{KGen}(\mathsf{par})$, if one executes n instances of protocol MSign, where the i-th instance executes on inputs (par, m, sk_i), and if all the messages between these instances are correctly delivered, then each instance outputs the same string σ *s.t.* $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ...pk_n\}, \sigma) = 1$.

Multisignature security in Key Verification model. As in the previous works on multisignatures, *e.g.* [MOR01, BN06, RY07], we define multisignature security as universal unforgeability under a chosen message attack against a single honest player. Namely, we define the *adversarial advantage* of an adversary $\mathcal{A}$ against the multisignature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{KVrfy}, \mathsf{MSign}, \mathsf{Vrfy})$, *i.e.* $\mathbf{Adv}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$, as a probability that experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$ described in Figure 2 outputs 1,where the probability goes over the random coins of the adversary $\mathcal{A}$ and all the randomness used in the experiment. We call a multisignature scheme $(t, \epsilon, n_{max}, q_s)$-secure if $\mathbf{Adv}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A}) \leq \epsilon$ for every adversary $\mathcal{A}$ that runs in time at most t, makes at most q_s signature queries, and where the size of the group of players S on behalf of which the adversary forges is bounded as $|S| \leq n_{max}$. In the random oracle model we consider also a notion of $(t, \epsilon, n_{max}, q_s, q_h)$-secure multisignature scheme, where adversary $\mathcal{A}$ is additionally restricted to at most q_h hash queries and the probability in the experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$ is taken also over the random choice of all hash functions.

Experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}(\mathcal{A})$
$\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$; $(sk^*, pk^*, \pi^*) \leftarrow \mathsf{KGen}(\mathsf{par})$; $\mathsf{List} \leftarrow \emptyset$;
Run $\mathcal{A}(\mathsf{par}, pk^*, \pi^*)$, and for every signature query m made by $\mathcal{A}$ do the following:
$\mathsf{List} \leftarrow \mathsf{List} \cup \{m\}$; Execute protocol MSign on behalf of an honest player on inputs (par, m, sk^*), forwarding messages to and from $\mathcal{A}$.
When $\mathcal{A}$ halts, parse its output as $(m, \sigma, \{(pk_i, \pi_i)\}_{i \in S})$ where S is some set of indexes.
If $(m \notin \mathsf{List})$ and $(pk_1 = pk^*)$ and $(\mathsf{KVrfy}(\mathsf{par}, pk_i, \pi_i) = 1$ for all $i \in \mathcal{S})$ and finally $(\mathsf{Vrfy}(\mathsf{par}, m, \{pk_i\}_{i \in S}, \sigma)) = 1$ then return 1; Otherwise return 0.

Fig. 2. Chosen Message Attack against Multisignature Scheme

Remarks on MS syntax and definition of security: (1) In the security experiment $\mathbf{Exp}^{\mathsf{uu-cma}}_{\mathsf{MS}}$ above we take the simplifying assumption that the Setup procedure is executed by an honest party. However, the public parameters in the two multisignature schemes in this paper are needed to define groups of prime order where the CDH and DDH assumptions hold, and such parameters can be chosen by a potentially dishonest party and then verified by every player. **(2)** The syntax of a multisignature scheme in the KV model is a simplification of the syntax used by [RY07], which modeled potentially interactive key registration processes. Here we allow only *non-interactive* proofs, but such proofs make multisignature schemes more flexible because they can be verified either by the CA's during the key registration process, as in [RY07], or by multisignature verifiers, e.g. together with verification of PKI certificates for a given key. **(3)** Note that a multisignature in the KV model generalizes multisignatures in the plain-model if one sets the proofs of public key validity to empty strings and sets the output of KVrfy on any inputs to 1. **(4)** However, in contrast to the definition of multisignatures in the plain model proposed by [MOR01] and [BN06], we do not include the set of participants' identities and/or their public keys as input in the multisignature protocol. The participating players must be aware of one another in the protocol execution, but this information is needed only to ensure proper communication, and does not need to be part of the inputs to the multisignature protocol. Removing this input from the multisignature protocol gives more flexibility to applications of multisignatures, because in some applications signers might care only about the message they are is signing and not about the identities of other signers. This is the case for example in aggregation of acknowledgments of broadcast reception. In such applications multisignature schemes analyzed in the model of [MOR01, BN06] would have to be preceded by an additional communication round for participants to broadcast their identities and/or public keys. On the other hand, multisignature schemes which conform to our simplified model imply schemes in the model of [MOR01, BN06] if the MSign protocol is executed on the message appended with the list of identities and/or public keys of the participating players. **(5)** The notion of multisignature security in [MOR01, BN06] treats a multisignature effectively as a signature on a *pair* (m, PKSet), and their notion of forgery is consequently broader than ours since it includes a case where an attacker forges a multisignature on a message that was previously signed by the honest player, but it was signed together with a different set of public keys. In our model such adversary would not be considered a successful forger since in our model an honest player is not required to be aware of the other participants in a multisignature protocol. However, a scheme secure according to our notion implies a scheme secure in the model of [MOR01, BN06] if players execute the MSign protocol on the concatenation of message m and the set of public keys PKSet of the participating players, as in item (4) above.

4 Three-Round DDH-Based Multisignature Scheme

We describe a multisignature scheme denoted MDDH, presented in Figure 3, with a tight security reduction from the DDH problem. The MDDH scheme is a multisignature version of the signature scheme of Katz and Wang [KW03], which also has an exact security reduction from the DDH problem. The MDDH scheme takes three rounds and

1. $\mathsf{Setup}(1^\kappa)$: Let G be a multiplicative group of prime order q and let g be a generator of G. Consider the following hash functions: $\mathcal{G}_1 : G^6 \rightarrow \mathbb{Z}_q$, $\mathcal{H}_1 : G \rightarrow \{0,1\}^{2\kappa}$ and $\mathcal{H}_2 : G^6 \times \{0,1\}^* \rightarrow \mathbb{Z}_q$. Pick $g, h \xleftarrow{r} G$. Set $\mathsf{par} \leftarrow (g, h, q)$.
2. $\mathsf{KGen}(\mathsf{par})$: Player P_i picks his (sk_i, pk_i, π_i) tuple as follows:
   Pick $x_i \xleftarrow{r} \mathbb{Z}_q$, compute $y_i \leftarrow g^{x_i}$, $z_i \leftarrow h^{x_i}$ and set $pk_i \leftarrow (y_i, z_i)$, $sk_i \leftarrow x_i$;
   Construct a "proof of possession" of x_i, *i.e.* a NIZK proof of $\mathsf{DL}_g(y_i) = \mathsf{DL}_h(z_i)$:
   Pick $k \xleftarrow{r} \mathbb{Z}_q$ and compute $u \leftarrow g^k$, $v \leftarrow h^k$;
   Set $e \leftarrow \mathcal{G}_1(g, h, y_i, z_i, u, v)$, compute $s \leftarrow k + ex_i$ and set $\pi_i \leftarrow (s, e)$.
3. $\mathsf{KVrfy}(\mathsf{par}, pk, \pi)$: Let pk=(y, z), π=(s, e). Accept if $e = \mathcal{G}_1(g, h, y, z, g^s y^{-e}, h^s z^{-e})$
4. Protocol MSign: Let $\mathcal{S}$ where $\mathcal{S} \leq n_{max}$ be the set of players that participate in the protocol. Each player can determine the set $\mathcal{S}$ after the first protocol step. Player P_i on inputs (par, m, sk_i), performs the following steps:
   4.1 Pick $k_i \xleftarrow{r} \mathbb{Z}_q$, compute $A_i \leftarrow g^{k_i}$, $B_i \leftarrow h^{k_i}$, $C_{A_i} \leftarrow \mathcal{H}_1(A_i)$, $C_{B_i} \leftarrow \mathcal{H}_1(B_i)$;
   Broadcast $(y_i, z_i, C_{A_i}, C_{B_i})$.
   4.2 Receive (C_{A_j}, C_{B_j}) for all $P_j \in \mathcal{S}/\{P_i\}$ and broadcast (A_i, B_i).
   4.3 Receive (A_j, B_j, y_j, z_j) for all $P_j \in \mathcal{S}/\{P_i\}$;
   Abort if $C_{A_j} \neq \mathcal{H}_1(A_j)$ or $C_{B_j} \neq \mathcal{H}_1(B_j)$ for any $P_j \in \mathcal{S}/\{P_i\}$;
   Compute $y \leftarrow \prod_{P_j \in \mathcal{S}} y_j$, $z \leftarrow \prod_{P_j \in \mathcal{S}} z_j$, $A \leftarrow \prod_{P_j \in \mathcal{S}} A_j$, $B \leftarrow \prod_{P_j \in \mathcal{S}} B_j$;
   Set $e \leftarrow \mathcal{H}_2(g, h, y, z, A, B, m)$, compute $s_i \leftarrow ex_i + k_i$ and broadcast s_i.
   4.4 Output multisignature $\sigma = (e, s)$, where $s = \sum_{P_j \in \mathcal{S}} s_j$.
5. $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ..., pk_n\}, \sigma)$:
   Parse σ as (e, s) and each pk_i as (y_i, z_i). Compute $y \leftarrow \prod_{i=1}^n y_i$ and $z \leftarrow \prod_{i=1}^n z_i$;
   If $e = \mathcal{H}_2(g, h, y, z, g^s y^{-e}, h^s z^{-e}, m)$ then accept otherwise reject.

Fig. 3. MDDH multisignature scheme

has fast signing and verification procedures. It requires only two group exponentiations per party for signing and two double-exponentiations for verification. The length of the MDDH signature is $2|q|$, which can be 320 bits, only twice the size of shortest multisignature schemes [RY07, BGLS03], which, however, are based on a potentially stronger assumption of GapDH on a group with a bilinear map.

Theorem 1. *If DDH problem is (t', ϵ')-hard in group G, then multisignature* MDDH *in Figure 3 is $(t, \epsilon, n_{max}, q_s, q_h)$-secure in ROM where*

$$\epsilon \leq \epsilon' + \frac{{q_h}^2 + 2q_s(q_h + n_{max})}{2^{2\kappa}} + \frac{q_s q_h}{q - q_h} + \frac{q_h}{q}$$
$$t \geq t' - 2.4(q_s + n_{max})t_e - 4q_s n_{max} t_m$$

and t_m and t_e are the times of one multiplication and one q-bit exponentiation in G.

Proof sketch. Due to space constraints we relegate the formal proof of this theorem to the full version of this paper [BJ08], but we give a rough sketch of the proof here. Given an adversary $\mathcal{A}$ against the MDDH scheme we construct an algorithm $\mathcal{B}$ that solves the DDH problem in group G as follows: The reduction embeds its DDH challenge (g, h, y_1, z_1) into the public key of the sole honest player P_1 by setting $\mathsf{par} = (g, h)$ and $pk_1 = (y_1, z_1)$. Note that the multisignature protocol performed by each player

is a version of a standard HVZK proof system for DL-equality, with the first message A_i, B_i in this proof system preceded by a round of ROM-based commitments C_{A_i}, C_{B_i}. As observed in [BN06], this round of commitment enables straight-line simulation of this HVZK proof system: Namely, $\mathcal{B}$ picks both the challenge e and P_1's response s_1 uniformly at random in $\mathbb{Z}_q$, computes $A_1 = g^{s_1} y_1^{-e}$ and $B_1 = h^{s_1} z_1^{-e}$, *pretends* to commit to these values by sending random C_{A_1}, C_{B_1} values, and thanks to the fact that the adversary commits to his contributions A_i and B_i for $P_i \in \mathcal{S}/\{P_1\}$, the reduction $\mathcal{B}$ can compute the values A and B before she publishes his own contribution A_1, B_1 in step 4.2. In this way $\mathcal{B}$ can embed the challenge e in the output to the appropriate query (g, h, y, z, A, B, m) made by $\mathcal{A}$ to $\mathcal{H}_2$. This reduction fails only if (1) $\mathcal{A}$ manages to change one of his committed values; (2) $\mathcal{A}$ manages to decommit any of his commitments C to some X s.t. $C = \mathcal{H}_1(X)$ without querying $\mathcal{H}_1$ on X; (3) $\mathcal{A}$ makes query (g, h, y, z, A, B, m) to $\mathcal{H}_2$ before the reduction embeds challenge e in the answer. However, all these cases happen with at most negligible probability in ROM. Finally, the special soundness property of the above proof system ensures that if both the multisignature verification and all the key verification procedures pass then both (g, h, y, z) where $y = \prod_{i \in \mathcal{S}}(y_i)$ and $z = \prod_{i \in \mathcal{S}}(x_i)$ and (g, h, y_i, z_i) for each P_i are DH tuples, and hence so must be the input tuple (g, h, y_1, z_1). We leave the details of this proof to the full version [BJ08].

5 Three-Round CDH-Based Multisignature Scheme

We describe a multisignature scheme, denoted MCDH and presented in Figure 4, whose security is tightly related to the *expected-time* hardness of the CDH problem in the Key Verification model. The MCDH scheme is a multisignature version of the CDH-based signature of [KW03] and [GJ03]. It takes three rounds, the multisignature is $(|G| + 2|q| + 2\kappa)$-bit long, and its verification procedure requires only two exponentiations. We note, however, that the low round complexity and a tight reduction from the (expected time) CDH problem comes at the following non-standard cost: Each player in the multisignature generation procedure must verify ZK proofs issued by the other players, and thus the computational cost of the signing algorithm is $O(n)$ exponentiations where n is the size of the signing group. This, however, will not be an important drawback as long as the number of signers is modest or if the communication costs in n-sized group of signers dominate the $O(n)$-exponentiations cost for each signer. As we discuss at the end of this section, this cost can be avoided if one tolerates an increase in the number of protocol rounds.

Theorem 2. *If there is no adversary that can solve the CDH problem in group G in expected time t' with probability ϵ', then the multisignature scheme* MCDH*, described in figure 4 is $(t, \epsilon, n_{max}, q_s, q_h)$-secure in random oracle model where*

$$\epsilon \leq 2\epsilon' + \frac{2q_h^2 + 4q_s q_h}{2^{2\kappa}} + \frac{(2q_s + 3)q_h}{q} + \frac{q_s q_h}{q - q_h}$$

$$t \geq \frac{1}{2}\left(t' - (q_h - 6(n_{max} + 1)(q_s + 1))t_e - 4q_s n_{max} t_m\right)$$

where t_m and t_e are the times of one multiplication and one q-bit exponentiation in G.

1. $\mathsf{Setup}(1^\kappa)$: Let G be a multiplicative group of prime order q and let g be a generator of G. Consider the following hash functions: $\mathcal{G}_1 : G \to G$, $\mathcal{G}_2 : G^6 \to \mathbb{Z}_q$, $\mathcal{H}_1 : G \to \{0,1\}^{2\kappa}$, $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^{2\kappa}$, $\mathcal{H}_3 : \{0,1\}^* \times \{0,1\}^{2\kappa} \to G$, $\mathcal{H}_4 : G^8 \to \mathbb{Z}_q^2$ and $\mathcal{H}_5 : G^6 \to \mathbb{Z}_q$. Set $\mathsf{par} \leftarrow (g, q)$;
2. $\mathsf{KGen}(\mathsf{par})$: Player P_i picks an (sk_i, pk_i, π_i) tuple as follows:
   Pick $x_i \xleftarrow{r} \mathbb{Z}_q$, compute $y_i \leftarrow g^{x_i}$ and set $pk_i \leftarrow y_i$ and $sk_i \leftarrow x_i$;
   Construct π_i as "proof of possession" of x_i:
   Set $h \leftarrow \mathcal{G}_1(y_i)$, $z \leftarrow h^{x_i}$ and construct a NIZK proof of $\mathsf{DL}_g(y_i) = \mathsf{DL}_h(z)$:
   Pick $k \xleftarrow{r} \mathbb{Z}_q$ and compute $u \leftarrow g^k$, $v \leftarrow h^k$;
   Set $e \leftarrow \mathcal{G}_2(g, h, y_i, z, u, v)$, compute $s \leftarrow k + ex_i$ and set $\pi_i \leftarrow (z, s, e)$;
3. $\mathsf{KVrfy}(\mathsf{par}, pk, \pi)$: Let $pk = y$, $\pi = (z, s, e)$ and $h \leftarrow \mathcal{G}_1(y)$.
   Accept if $e = \mathcal{G}_2(g, h, y, z, g^s y^{-e}, h^s z^{-e})$.
4. Protocol MSign: Let $\mathcal{S}$ where $\mathcal{S} \leq n_{max}$ be the set of players that participate in the protocol. Each player can determine the set $\mathcal{S}$ after the first protocol step. Player P_i on inputs (par, m, sk_i), performs the following steps:
   4.1 Pick $k_i \xleftarrow{r} \mathbb{Z}_q$, compute $A_i \leftarrow g^{k_i}$ and set $C_{A_i} \leftarrow \mathcal{H}_1(A_i)$. If bit $b_i^{(m)}$ is not set, pick $b_i^{(m)} \xleftarrow{r} \{0,1\}$. Broadcast $(b_i^{(m)}, y_i, C_{A_i})$.
   4.2 Receive $(b_j^{(m)}, y_j, C_{A_j})$ for all $P_j \in \mathcal{S}/\{P_i\}$; Abort if for any of them, $C_{A_j} = C_{A_i}$.
   Set $p \leftarrow \mathcal{H}_2(b_1^{(m)} | b_2^{(m)} | ... | b_{|\mathcal{S}|}^{(m)})$, where the global order among players can be determined *e.g.* by hashing the message broadcasted by players in the previous round. (If two players broadcast the same message, the order between them can be arbitrary.)
   Set $h \leftarrow \mathcal{H}_3(m, p)$ and Compute $z_i \leftarrow h^{x_i}$ and $B_i \leftarrow h^{k_i}$;
   Construct NIZK proof π_i that $\mathsf{DL}_g(y_i) = \mathsf{DL}_h(z_i)$ and $\mathsf{DL}_g(A_i) = \mathsf{DL}_h(B_i)$:
   Pick $r \xleftarrow{r} \mathbb{Z}_q$, set $u \leftarrow g^r$, $v \leftarrow h^r$ and $(e_i, f_i) \leftarrow \mathcal{H}_4(g, h, y_i, z_i, A_i, B_i, u, v)$;
   Compute $t_i \leftarrow r + e_i x_i + f_i k_i$ and set $\pi_i \leftarrow (e_i, f_i, t_i)$;
   Broadcast (A_i, B_i, z_i, π_i).
   4.3 Receive (A_j, B_j, z_j, π_j) for all $P_j \in \mathcal{S}/\{P_i\}$; Let $\pi_j = (e_j, f_j, t_j)$;
   Abort if $(e_j, f_j) \neq \mathcal{H}_4(g, h, y_j, z_j, A_j, B_j, g^{t_j} y_j^{-e_j} A_j^{-f_j}, h^{t_j} z_j^{-e_j} B_j^{-f_j})$ or $C_{A_j} \neq \mathcal{H}_1(A_j)$ for any $P_j \in \mathcal{S}/\{P_i\}$.
   Compute $y \leftarrow \prod_{P_j \in \mathcal{S}} y_j$, $z \leftarrow \prod_{P_j \in \mathcal{S}} z_j$, $A \leftarrow \prod_{P_j \in \mathcal{S}} A_j$ and $B \leftarrow \prod_{P_j \in \mathcal{S}} B_j$;
   Set $e \leftarrow \mathcal{H}_5(g, h, y, z, A, B)$, $s_i \leftarrow k_i + ex_i$ and broadcast s_i.
   4.4 Output $\sigma = (z, e, s, p)$, where $s = \sum_{P_j \in \mathcal{S}} s_j$.
5. $\mathsf{Vrfy}(\mathsf{par}, m, \{pk_1, pk_2, ..., pk_n\}, \sigma)$:
   Parse σ as (z, e, s, p) and each pk_i as y_i. Set $y \leftarrow \prod_{i=1}^n y_i$ and $h \leftarrow \mathcal{H}_3(m, p)$;
   If $e = \mathcal{H}_5(g, h, y, z, g^s y^{-e}, h^s z^{-e})$ then accept otherwise reject.

Fig. 4. MCDH multisignature scheme

Proof. Let $\mathcal{A}$ be an adversary that attacks the multisignature scheme MCDH, depicted in figure 4, in time t and with success probability ϵ and makes q_s signing queries and at most q_h hash queries and produces a forgery on behalf of $n \leq n_{max}$ players. We construct an algorithm $\mathcal{B}$, that given oracle access to $\mathcal{A}$, solves CDH problem in group G, *i.e.* given $(g, y_1, \hat{h}) \in G^3$ where $y_1 = g^{x_1}$ it outputs $\hat{h}^{x_1}$ in expected time t' and with success probability ϵ'. Assume $\mathcal{A}$ makes $q_{\mathcal{G}_1}$, $q_{\mathcal{G}_2}$, $q_{\mathcal{H}_1}$, $q_{\mathcal{H}_2}$, $q_{\mathcal{H}_3}$, $q_{\mathcal{H}_4}$ and $q_{\mathcal{H}_5}$ queries to $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, $\mathcal{H}_4$ and $\mathcal{H}_5$ respectively and $q_{\mathcal{G}_1} + q_{\mathcal{G}_2} + q_{\mathcal{H}_1} + q_{\mathcal{H}_2} + q_{\mathcal{H}_3} + q_{\mathcal{H}_4} + q_{\mathcal{H}_5} \leq q_h$, the total number of hash queries. In what follows we show how the CDH-attacker $\mathcal{B}$ proceeds given a CDH challenge $(g, y_1, \hat{h})$.

Initialization: The algorithm $\mathcal{B}$ sets up tables G_1, G_2, H_1, H_2, H_3, H_4 and H_5 to simulate the hash functions $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$, $\mathcal{H}_4$ and $\mathcal{H}_5$ respectively which are filled out throughout the simulation. It also uses table B to store the bits $b^{(m)}$ assigned to each message m. The algorithm $\mathcal{B}$ then sets the public parameters and the honest player's public key as $\mathsf{par} = g$ and $pk_1 = y_1$ respectively. Algorithm $\mathcal{B}$ picks $\beta_{y_1} \xleftarrow{r} \mathbb{Z}_q$ and assigns $\mathsf{G}_1[y_1] = g^{\beta_{y_1}}$. It then computes $z_1 \leftarrow (y_1)^{\beta_{y_1}}$ and produces a simulated NIZK proof of DL-equality between $\mathsf{DL}_g(y_1)$ and $\mathsf{DL}_h(z_1)$ where $h = \mathsf{G}_1[y_1]$. To simulate this proof, $\mathcal{B}$ picks $e, s \xleftarrow{r} \mathbb{Z}_q$ and assigns $\mathsf{G}_2[(g, h, y_1, z_1, g^s y_1^{-e}, h^s z_1^{-e})] = e$. Finally, $\mathcal{B}$ executes $\mathcal{A}$ on input $(\mathsf{par}, pk_1, \pi_1)$ where $\pi_1 = (s, e)$. Note that $\mathsf{G}_1[y_1]$ and $\mathsf{G}_2[(g, h, y_1, z_1, g^s y_1^{-e}, h^s z_1^{-e})]$ are set before the execution of $\mathcal{A}$ starts. Note also that indeed $\mathsf{DL}_g(y_1) = \mathsf{DL}_h(z_1)$ since $z_1 = (y_1)^{\beta_{y_1}} = g^{x_1 \beta_{y_1}} = h^{x_1}$.

Answering hash queries: If a query has been made before, then $\mathcal{B}$ returns the appropriate entry from the appropriate table *e.g.* if $\mathcal{A}$ queries $\mathcal{G}_1$ on y, $\mathcal{B}$ responds with $\mathsf{G}_1[y]$. If $\mathcal{A}$ makes a new query to $\mathcal{G}_2$, $\mathcal{H}_1$, $\mathcal{H}_4$ and $\mathcal{H}_5$, $\mathcal{B}$ answers with an element chosen uniformly at random from the appropriate domain. If $\mathcal{A}$ makes a new query y to $\mathcal{G}_1$, $\mathcal{B}$ answers with $\hat{h}^{\beta_y}$ where $\beta_y \xleftarrow{r} \mathbb{Z}_q$. If $\mathcal{A}$ makes a new query $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ to $\mathcal{H}_2$, $\mathcal{B}$ answers with an element chosen uniformly at random form $\{0,1\}^{2\kappa}$ except when the following failure cases happen: (a) If a collision happens in $\mathcal{H}_2$ then the simulator sets a flag $\mathsf{bad}_1 \leftarrow \mathsf{true}$, stops and returns "fail". (b) If $\mathcal{A}$ queries $\mathcal{H}_2$ for the first time on $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ and $\mathcal{H}_2(\boldsymbol{b}) = p$ for some previous query (m, p) to $\mathcal{H}_3$ then the simulator sets a flag $\mathsf{bad}_2 \leftarrow \mathsf{true}$, stops and returns "fail". If $\mathcal{A}$ makes a new query (m, p) to $\mathcal{H}_3$, the simulator looks up the table H_2: (a) If there exists no entry in H_2 corresponding to p then the simulator picks $\alpha_{(m,p)} \xleftarrow{r} \mathbb{Z}_q$ and answers the query as $\hat{h}^{\alpha_{(m,p)}}$; (b) If the simulator finds an entry $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ corresponding to p in H_2 then it assigns a bit ${b_1}^{(m)}$ to message m if it has not been yet assigned, picks $\alpha_{(m,p)} \xleftarrow{r} \mathbb{Z}_q$ and checks whether the first element of the vector $\boldsymbol{b}$ is equal to ${b_1}^{(m)}$: If so, the simulator responds to the query as $g^{\alpha_{(m,p)}}$ and otherwise it responds to the query as $\hat{h}^{\alpha_{(m,p)}}$.

Answering signature queries: To answer each signature query, simulator runs the adversary twice after step 4.1. In the first execution, $\mathcal{B}$ runs the adversary to the point that it can learn A_j, B_j and z_j for all $P_j \in \mathcal{S}$, but it does not complete this execution since it will not know how to correctly create the response s_1 in the zero-knowledge proof on behalf of player P_1. The simulator then rewinds the adversary and uses the values it has learned in the first execution to simulate the proof-of-knowledge protocol on behalf of player P_1. If the adversary uses the same values $\{A_j, B_j, z_j\}_{P_j \in \mathcal{S}}$ in both executions then the simulator knows the query $\mathcal{H}_5\,(g, h, y, z, A, B)$ into which it should embed its chosen challenge e. The only way this simulation can fail is if the adversary changes his values z_j, A_j and B_j for $P_j \in \mathcal{S}/\{P_1\}$ in the second execution. However due to the soundness property of NIZK proof of DL-equality and the collision resistance property of $\mathcal{H}_1$, this can happen with only a negligible probability. This is because the adversary has revealed y_j's and committed to A_j's in the first round. Moreover, the adversary gives a NIZK proof that $\mathsf{DL}_g(y_j) = \mathsf{DL}_h(z_j)$ and $\mathsf{DL}_g(A_j) = \mathsf{DL}_h(B_j)$ for all $P_j \in \mathcal{S}/\{P_1\}$. The details of the signature query simulation on input m are given bellow. We point out that if the adversary aborts and/or sends values which do not pass

verification procedures in any point in the simulation below then the simulator stops this instance of the multisignature generation protocol, just like an honest player P_1.

Step 1. If bit $b_1^{(m)}$ has not been chosen for message m, pick $b_1^{(m)} \stackrel{r}{\leftarrow} \{0,1\}$; Otherwise use the previously chosen value. Pick $C_{A_1} \stackrel{r}{\leftarrow} \{0,1\}^{2\kappa}$. Send $(b_1^{(m)}, y_1, C_{A_1})$ to $\mathcal{A}$.

Step 2,3. Upon receiving $(b_j^{(m)}, y_j, C_{A_j})$ for all $P_j \in \mathcal{S}/\{P_1\}$, verify whether for all $P_j \in \mathcal{S}/\{P_1\}$, $C_{A_j} \neq C_{A_1}$; Abort if verification fails. Set $p \leftarrow \mathcal{H}_2(b_1^{(m)}|b_2^{(m)}|...|b_{|\mathcal{S}|}^{(m)})$ and $h \leftarrow \mathcal{H}_3(m,p)$. Retrieve $\alpha_{(m,p)}$ assigned to (m,p) in the simulation of this query to $\mathcal{H}_3$ and compute $z_1 \leftarrow y_1^{\alpha_{(m,p)}}$. Note that our simulation procedure for $\mathcal{H}_3$ ensures that $h = \mathcal{H}_3(m,p) = g^{\alpha_{(m,p)}}$ and thus $z_1 = y_1^{\alpha_{(m,p)}} = g^{x_1\alpha_{(m,p)}} = h^{x_1}$.

Run SIM_R as a subroutine and let $\{(A_j^{(1)}, B_j^{(1)}, z_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}}$ be the values it returns. If SIM_R did not stop, rewind the adversary to the point where SIM_R is called, and run SIM_L as a subroutine on input $\{(A_j^{(1)}, B_j^{(1)}, z_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}}$.

Step 4. Compute the multisignature from appropriate values gained in SIM_R simulation.

Procedure SIM_R:

Step 2'. Pick $k_1 \stackrel{r}{\leftarrow} \mathbb{Z}_q$ and compute $A_1^{(1)} \leftarrow g^{k_1}$, $B_1^{(1)} \leftarrow h^{k_1}$. If $\mathsf{H}_1[A_1^{(1)}]$ is not set, assign $\mathsf{H}_1[A_1^{(1)}] \leftarrow C_{A_1}$; Otherwise set $\mathsf{bad}_3 \leftarrow \mathsf{true}$, stop and return "fail".

Simulate the NIZK proof that $\mathsf{DL}_g(y_1) = \mathsf{DL}_h(z_1)$ and $\mathsf{DL}_g(A_1^{(1)}) = \mathsf{DL}_h(B_1^{(1)})$:

Pick $e_1, f_1, t_1 \stackrel{r}{\leftarrow} \mathbb{Z}_q^3$, set $u_1 \leftarrow g^{t_1} y_1^{-e_1} (A_1^{(1)})^{-f_1}$, $v_1 \leftarrow h^{t_1} z_1^{-e_1} (B_1^{(1)})^{-f_1}$.

If $\mathsf{H}_4[(g,h,y_1,z_1,A_1^{(1)},B_1^{(1)},u_1,v_1)]$ is not set, set it to (e_1,f_1); Otherwise set $\mathsf{bad}_4 \leftarrow \mathsf{true}$, stop and return "fail".

Send $(A_1^{(1)}, B_1^{(1)}, z_1, (e_1,f_1,t_1))$ to $\mathcal{A}$.

Step 3'. Upon receiving $(A_j, B_j, z_j, (e_j, f_j, t_j))$ for all $P_j \in \mathcal{S}/\{P_1\}$, verify whether for all $P_j \in \mathcal{S}/\{P_1\}$, $(e_j, f_j) = \mathcal{H}_4(g,h,y_j,z_j,A_j,B_j, g^{t_j} y_j^{-e_j} A_j^{-f_j}, h^{t_j} z_j^{-e_j} B_j^{-f_j})$ and $C_{A_j} = \mathcal{H}_1(A_j)$. If the verification does not pass, stop the simulation of this multisignature instance. Otherwise return $\{(A_j, B_j, z_j)\}_{P_j \in \mathcal{S}/\{P_1\}}$.

Procedure $SIM_L(\{(z_j^{(1)}, A_j^{(1)}, B_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}})$:

Step 2. Pick $(s_1, e) \stackrel{r}{\leftarrow} \mathbb{Z}_q^2$ and compute $A_1^{(2)} \leftarrow g^{s_1} y_1^{-e}$, $B_1^{(2)} \leftarrow h^{s_1} z_1^{-e}$. If $\mathsf{H}_1[A_1^{(2)}]$ is not set, assign $\mathsf{H}_1[A_1^{(2)}] \leftarrow C_{A_1}$; Otherwise set $\mathsf{bad}_5 \leftarrow \mathsf{true}$, stop and return "fail".

Simulate the NIZK proof that $\mathsf{DL}_g(y_1) = \mathsf{DL}_h(z_1)$ and $\mathsf{DL}_g(A_1^{(2)}) = \mathsf{DL}_h(B_1^{(2)})$

Pick $e_1, f_1, t_1 \stackrel{r}{\leftarrow} \mathbb{Z}_q^3$, set $u_1 \leftarrow g^{t_1} y_1^{-e_1} (A_1^{(2)})^{-f_1}$, $v_1 \leftarrow h^{t_1} z_1^{-e_1} (B_1^{(2)})^{-f_1}$.

If $\mathsf{H}_4[(g, h, y_1, z_1, A_1^{(2)}, B_1^{(2)}, u_1, v_1)]$ is not set, set it to (e_1, f_1); Otherwise set $\mathsf{bad}_6 \leftarrow \mathsf{true}$, stop and return "fail".

Compute $y \leftarrow \prod_{P_j \in \mathcal{S}} y_j$, $z \leftarrow z_1 \prod_{P_j \in \mathcal{S}/\{P_1\}} z_j^{(1)}$, $A \leftarrow A_1^{(2)} \prod_{P_j \in \mathcal{S}/\{P_1\}} A_j^{(1)}$, $B \leftarrow B_1^{(2)} \prod_{P_j \in \mathcal{S}/\{P_1\}} B_j^{(1)}$. If $\mathsf{H}_5[(g, h, y, z, A, B)]$ is not set, set it to e; Otherwise set $\mathsf{bad}_7 \leftarrow \mathsf{true}$, stop and return "fail".

Send $(A_1^{(2)}, B_1^{(2)}, z_1, (e_1, f_1, t_1))$ to $\mathcal{A}$.

Step 3. Upon receiving $(A_j, B_j, z_j, (e_j, f_j, t_j))$ for all $P_j \in \mathcal{S}/\{P_1\}$, verify whether for all $P_j \in \mathcal{S}/\{P_1\}$,

(a) $A_j = A_j^{(1)}$; If not, set $\mathsf{bad}_8 \leftarrow \mathsf{true}$, stop and return "fail".

(b) $B_j = B_j^{(1)}$ and $z_j = z_j^{(1)}$; If not, set $\mathsf{bad}_9 \leftarrow \mathsf{true}$, stop and return "fail".

(c) $(e_j, f_j) = \mathcal{H}_4(g, h, y_j, z_j, A_j, B_j, g^{t_j} y_j^{-e_j} A_j^{-f_j}, h^{t_j} z_j^{-e_j} B_j^{-f_j})$ and $C_{A_j} = \mathcal{H}_1(A_j)$; If not stop the simulation of this multisignature instance.

If all the verifications pass, send s_1 to $\mathcal{A}$.

Finalization: After receiving a valid forgery $(m, \sigma, \{(pk_i, \pi_i)\}_{P_i \in \mathcal{S}})$ from $\mathcal{A}$, the algorithm $\mathcal{B}$ attempts to output $\hat{z} = \hat{h}^{x_1}$. Let $\sigma = (z, e, s, p)$ and $pk_i = y_i$ and $\pi_i = (z_i, e_i, s_i)$ for all $P_i \in \mathcal{S}$. If in the simulation of $\mathcal{H}_3$ for query (m, p), the simulator finds an entry $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ corresponding to p in H_2, and moreover the first element of the vector $\boldsymbol{b}$ is equal to $b_1^{(m)}$, then it stops and returns "fail"; otherwise $\mathcal{B}$ retrieves $\alpha_{(m,p)}$ assigned to (m, p) in the simulation of $\mathcal{H}_3$ and β_{y_i} assigned to y_i in the simulation to $\mathcal{G}_1$ for all $P_i \in \mathcal{S}$ and returns $\hat{z}$ where

$$\hat{z} = \begin{cases} \frac{z^{1/\alpha_{(m,p)}}}{\prod_{P_i \in \mathcal{S}/\{1\}} (z_i)^{1/\beta_{y_i}}} & \text{when } \mathcal{S}/\{P_1\} \neq \emptyset \\ z^{1/\alpha_{(m,p)}} & \text{otherwise} \end{cases}$$

Note that if $(g, y, \hat{h}^{\alpha_{(m,p)}}, z)$ where $y = \prod_{P_i \in \mathcal{S}} (y_i)$ and $(g, y_i, \hat{h}^{\beta_{y_i}}, z_i)$ where $P_i \in \mathcal{S}/\{P_1\}$ are all DH tuples, then $(g, y_1, \hat{h}, \hat{z})$ is also a DH tuple. We will argue that if the multisignature verification passes then with a high probability $(g, y, \hat{h}^{\alpha_{(m,p)}}, z)$ is a DH tuple and if the key verification passes for all of the adversary's public keys then with a high probability $(g, y_i, \hat{h}^{\beta_{y_i}}, z_i)$'s are also all DH tuples, in which case $\hat{z}$ is the answer to the reduction's CDH challenge. But first let's look at the probability of failure events. Let E_i for $i = 1..9$, denote the failure event that $\mathsf{bad}_i = \mathsf{true}$. The algorithm $\mathcal{B}$ provides a perfect simulation for adversary $\mathcal{A}$ conditioned on events E_i where $i = 1..9$ not happening. More precisely, if events E_i for $i = 1..9$ do not happen then: Firstly, the view of the adversary interacting with the simulator in SIM_R branch is identical to the view of the adversary in the real execution conditioned on the event that the simulation stops in step (3'), *i.e.* if $\mathcal{A}$'s responses in that step do not pass the verification procedure. Secondly, the view of the adversary interacting with the simulator in SIM_L branch is identical to the view of the adversary in the real execution conditioned on the event that $\mathcal{A}$'s responses in step (3') are correct, *i.e.* that values $\{(z_j^{(1)}, A_j^{(1)}, B_j^{(1)})\}_{P_j \in \mathcal{S}/\{P_1\}}$

which are input to SIM_L satisfy the verification condition. (Note that the SIM_L branch executes only under this condition.) We start by upper-bounding the probabilities of all the "fail" events:

The event E_1 corresponds to a collision in $\mathcal{H}_2$. Thus $\Pr[E_1] \leq (q_{\mathcal{H}_2})^2/2^{2\kappa}$. To upper bound E_2, it is enough to upper bound the event that $\mathcal{H}_2$ is queried on some $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ where $\mathcal{H}_2(\boldsymbol{b}) = p$ for some previous query (m, p) to $\mathcal{H}_3$. The probability that any query to $\mathcal{H}_2$ hits some p *s.t.* (m, p) is also queried to $\mathcal{H}_3$ is at most $q_{\mathcal{H}_3}/2^{2\kappa}$ and there are at most $q_{\mathcal{H}_2}$ queries to $\mathcal{H}_2$, thus $\Pr[E_2] \leq q_{\mathcal{H}_2} q_{\mathcal{H}_3}/2^{2\kappa}$. The events E_3 and E_5 reflect the possibility that $\mathcal{H}_1$ has been queried on A_1 before it is set by $\mathcal{B}$ in a particular signing query. Since no information about A_1 is revealed before it is set by $\mathcal{B}$, thus both of these events can be upper bounded by $q_s q_{\mathcal{H}_1}/2^{2\kappa}$. Similarly both E_4 and E_6 can be upper bounded by $q_s q_{\mathcal{H}_4}/q$. The event E_7 reflects the possibility that $\mathcal{H}_5$ has been queried on (g, h, y, z, A, B) before it is set by $\mathcal{B}$ in a particular signing query. Here we have two cases: either adversary has queried $\mathcal{H}_1$ on A_1 or B_1 in a particular signing query or he has not. In the first case which happens with probability at most $2q_{\mathcal{H}_1}/2^{2\kappa}$, the adversary knows both A and B and can easily query $\mathcal{H}_5$ on (g, h, y, z, A, B) before it is set by $\mathcal{B}$. In the second case $\mathcal{A}$ still has some information about A and B and can happen to query $\mathcal{H}_5$ on (g, h, y, z, A, B) with probability at most $q_{\mathcal{H}_5}/(q - q_{\mathcal{H}_1})$. Thus, $\Pr[E_7] \leq 2q_s q_{\mathcal{H}_1}/2^{2\kappa} + q_s q_{\mathcal{H}_5}/(q - q_{\mathcal{H}_1})$. The event E_8 corresponds to a collision in $\mathcal{H}_1$. Now since $C_{A_j} \neq C_{A_1}$ for all $P_j \in \mathcal{S}$, and C_{A_1} is the only output of $\mathcal{H}_1$ that is being manipulated by $\mathcal{B}$ in SIM_R and SIM_L, therefore with regards to value C_{A_j}, for any $P_j \in \mathcal{S}/\{P_1\}$, the hash function $\mathcal{H}_1$ remains collision resistant across these SIM_L and SIM_R executions. Thus the value $A_j^{(1)}$ revealed for C_{A_j} in SIM_R and value A_j revealed for C_{A_j} in SIM_L can be different with probability at most $(q_{\mathcal{H}_1})^2/2^{2\kappa}$. Hence $\Pr[E_8] \leq (q_{\mathcal{H}_1})^2/2^{2\kappa}$. The event E_9 reflects the possibility that $\mathcal{A}$ has cheated on at least one of the NIZK proofs in SIM_R or SIM_L branches. However due to special soundness property of this double-DL-equality proof system, for given tuple $(g, h, y_i, z_i, A_i, B_i)$ not satisfying double-DL-equality, for any $u_i, v_i \in G$, there's at most q different $(e, f) \in \mathbb{Z}_q$ pairs that satisfy the verification equation for some t. Therefore the probability of hitting such pair in $q_{\mathcal{H}_4}$ queries is bounded by $q_{\mathcal{H}_4}/q$. Thus $\Pr[E_9] \leq 2q_{\mathcal{H}_4}/q$.

There is also a possibility of failure in reduction after $\mathcal{A}$ outputs a valid forgery. Namely if in the simulation of $\mathcal{H}_3$ for query (m, p) where m is in the forgery, the simulator finds an entry $\boldsymbol{b} = b_1|...|b_{|\mathcal{S}|}$ corresponding to p in H_2, and moreover the first element of the vector $\boldsymbol{b}$ is equal to $b_1^{(m)}$, the query is answered by $g^{\alpha_{(m,p)}}$ and therefore is useless for the reduction. However since with probability $1/2$, the first element of the vector $\boldsymbol{b}$ is not equal to $b_1^{(m)}$, therefore with probability at least $1/2$ the reduction proceeds to output $\hat{z}$ after obtaining a valid forgery from $\mathcal{A}$.

Now since $\mathsf{Vrfy}(g, m, \{pk_i\}_{P_i \in \mathcal{S}}, \sigma) = 1$, thus due to special soundness property of DL-equality proof system, except for a probability of $q_{\mathcal{H}_5}/q$, $(g, y, \hat{h}^{\alpha_{(m,p)}}, z)$ is a DH tuple. Similarly Since $\mathsf{KVrfy}(\mathsf{par}, pk_i, \pi) = 1$ for all $P_j \in \mathcal{S}/\{P_1\}$, except for a probability of $q_{\mathcal{G}_2}/q$, the tuples $(g, y_j, \hat{h}^{\beta_{y_j}}, z_j)$ where $P_j \in \mathcal{S}/\{P_1\}$ are all DH tuples. Therefore, $(g, y_1, \hat{h}, \hat{z})$ is also a DH tuple except for these error probabilities, and thus $\mathcal{B}$ solves the DH problem with advantage $\epsilon' = 1/2(\epsilon - \mathsf{err})$ where

$$\mathsf{err} \leq \frac{q_{\mathcal{H}_2}(q_{\mathcal{H}_2}+q_{\mathcal{H}_3})+4q_s q_{\mathcal{H}_1}+(q_{\mathcal{H}_1})^2}{2^{2\kappa}}+\frac{2(q_s+1)q_{\mathcal{H}_4}+q_{\mathcal{H}_5}+q_{\mathcal{G}_2}}{q}+\frac{q_s q_{\mathcal{H}_5}}{q-q_{\mathcal{H}_1}}$$

The fact that $q_{\mathcal{G}_1}+q_{\mathcal{G}_2}+q_{\mathcal{H}_1}+q_{\mathcal{H}_2}+q_{\mathcal{H}_3}+q_{\mathcal{H}_4}+q_{\mathcal{H}_5} \leq q_h$ yields the desired result.

Calculating the running time of the simulator is little bit complicated due to the tree-like nature of its execution structure. Let's focus on "answering to the signature queries" part since it is the most time consuming part of the execution. Let $r_1|r_2|...|r_i$ be the randomness of the real execution of the protocol in which the randomness of the j^{th} signature query instance, $1 \leq j \leq i$, is r_j. Therefore $r_1|r_2|...|r_i$ determines the path of real execution up to i^{th} signature query. Accordingly let $t_i(r_1|r_2|...|r_i)$ be the running time of the real execution of the protocol in i^{th} round of answering signature queries. By assumption we have $\forall r_1, r_2, ..., r_{q_s} \sum_{i=1}^{q_s} t_i(r_1|r_2|...|r_i) = T$ where T is the total running time of the forger in "answering to the signature queries" part.

Let s_i^L and s_i^R be the running times of the simulator in SIM_L and SIM_R branches of the execution respectively that interact with the adversary in the i^{th} signature query. As we mentioned before the view of the adversary interacting with the simulator in SIM_R branch is similar to the view of the adversary in real execution of the protocol conditioned on the halting of the adversary except possibly with some negligible factor of failure. Similarly except for a negligible probability, the view of the adversary interacting with the simulator in SIM_L branch is similar to the view of the adversary in real execution. This can be stated more formally as follows: for all random coins of the adversary, $(A, SIM_R)_{\$\ SIM_R} = (A, P)_{\$\ P|\mathcal{A}\ \text{aborts}}$ and $(A, SIM_L)_{\$\ SIM_L} = (A, P)_{\$\ P}$. This means that there is a one to one correspondence between the randomness of each signature query in the real execution and the execution of the simulator both in SIM_L and in SIM_R. Therefore we can calculate the running time of the simulator by calculating the following:

$$T' = t_1(r_1^L) + t_1(r_1^R) + ... + t_{q_s}(r_1^L|r_2^L|...|r_{q_s-1}^L|r_{q_s}^L) + t_{q_s}(r_1^L|r_2^L|...|r_{q_s-1}^L|r_{q_s}^R)$$

Note that there exist random coins that lead to non-constant reduction: Consider an extreme case in which $\forall_{1\leq i\leq q_s-1}, t_i(r_1^L|r_2^L|...|r_i^L) = 0$ and $t_i(r_1^L|r_2^L|...|r_i^R) = T$ and $t_{q_s}(r_1^L|r_2^L|...|r_{q_s}^L) = t_{q_s}(r_1^L|r_2^L|...|r_{q_s}^L) = 0$. In this case $T' = q_s T$. However we can still get a tight bound on the *expected* running time of the simulator.

$$\begin{aligned} E[T'] &= \sum_{r_1^L, r_1^R, ..., r_{q_s}^L, r_{q_s}^R \leftarrow \{0,1\}^{2^{2q_s\kappa}}} \left(\frac{1}{2^{2q_s\kappa}} \sum_{i=1}^{q_s} \left(t_i(r_1^L|r_2^L|...|r_i^L) + t_i(r_1^L|r_2^L|...|r_i^R) \right) \right) \\ &= \left(t_{q_s}(r_1^L|r_2^L|...|r_{q_s}^R) + \sum_{i=1}^{q_s-1} t_i(r_1^L|r_2^L|...|r_i^L) \right) + \sum_{i=1}^{q_s} t_i(r_1^L|r_2^L|...|r_i^L) \\ &+ \sum_{i=1}^{q_s-1} \left(t_i(r_1^L|r_2^L|...|r_i^R) - t_i(r_1^L|r_2^L|...|r_i^L) \right) = 2T \end{aligned}$$

The last equation is because according to the definition the first two terms add up to $2T$ and for any function f defined on any domain D, $\sum_{x,y\in D^2} (f(x) - f(y)) = 0$.

There are two multi-exponentiations and one single exponentiation in initialization phase, one single exponentiation per each query to $\mathcal{H}_3$ and one single exponentiation per each query to $\mathcal{G}_1$. The reduction also makes three single exponentiations, at most

$4n_{max} + 2$ multi-exponentiations and at most $4n_{max}$ multiplications per each signing query in the signing phase, and $2(n-1)$ multi-exponentiations in KVrfy and two multi-exponentiations in Vrfy algorithms and n_{max} single exponentiations in finalization phase. Therefore, ignoring the cost of hash-table lookups, assuming that a two or three exponent multi-exponentiation take at most 25% more time than an exponentiation, the total running time of the algorithm $\mathcal{B}$ can be upper-bounded as

$$t' \leq 2t + (q_h + 6(n_{max}+1)(q_s+1))t_e + 4q_s n_{max} t_m$$

Note on the Security Reduction. There are two crucial tricks we use which allow us to compress the protocol to three rounds and maintain exact security reduction. First, we use the signature randomization technique introduced by Katz and Wang [KW03], which in the multisignature setting extends the signed message with n bits instead of just one. This idea allows to replace the $1/q_s$ factor encountered in the security reduction for the full-domain hash signature with a constant factor of $1/2$. However, to make the exact security reduction to go through, each player must have its own bit for the reduction to play with, and hence the signature size grows by exactly n bits. However we use an intermediate hash function to avoid this linear blowup and maintain constant signature size. Secondly, we use the ZK proofs to ensure that the adversary cannot manipulate values z_j and B_j provided by potentially corrupt players in the second round. Inclusion of such ZK proofs in the protocol fixes these values across instances of the same adversarial algorithm executing on the same inputs. This enables a very simple rewinding schedule in the simulation: The simulator attempts the execution once, learns the adversarial contributions z_j, B_j if the adversary reveals them accompanied with a correct ZK proof, rewinds the adversary, and equipped with the knowledge of the values the adversary is bound to use again (we are aided here by the fact that the soundness of the ZK proofs we use is unconditional), the simulator then successfully straight-line simulates the protocol on behalf of an honest player. If the adversary fails to reveal correct z_j, B_j values, the simulator has an even easier job because the first execution already forms a correct simulation, since the honest player would abandon the protocol if any protocol participant failed in this way. Thus the simulator repeats an execution of every instance of the signature scheme at most twice. At surface, the time of such simulation seems to be at most twice the total time of the adversary. However, upon closer inspection, it is clear that there are adversaries for which the running time of the simulator is q_s times the running time of the adversary. Namely in an extreme scenario in which the adversary takes its maximum time on the random coins that run it in the first execution in the simulation and takes zero time on the remaining random coins. However as we argue in the proof of the theorem 2 the expected running time of the simulation is at most twice as the expected running time of the adversary.

References

[BGLS03] Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)

[BJ08] Bagherzandi, A., Jarecki, S.: Multisignatures using proofs of secret key possession, as secure as the diffie-hellman problem. ePrint Archive (2008)

[BLS04] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptology 17(4), 297–319 (2004)

[BN06] Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: ACM Conference on Computer and Communications Security, pp. 390–399 (2006)

[BNN07] Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (2007)

[Bol03] Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2002)

[Bon98] Boneh, D.: The decision diffie-hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)

[Fis05] Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005)

[GJ03] Goh, E.-J., Jarecki, S.: A signature scheme as secure as the diffie-hellman problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 401–415. Springer, Heidelberg (2003)

[Har94] Harn, L.: Group-oriented (t,n) threshold digital signature scheme and digital multisignature. In: IEEE Proceedings on Computers and Digital Techniques, vol. 141(5), pp. 307–313 (1994)

[KW03] Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: ACM Conference on Computer and Communications Security, pp. 155–164 (2003)

[LHL94] Li, C.-M., Hwang, T., Lee, N.-Y.: Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 194–204. Springer, Heidelberg (1995)

[LOS+06] Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)

[MOR01] Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: extended abstract. In: ACM Conference on Computer and Communications Security, pp. 245–254 (2001)

[MW99] Maurer, U.M., Wolf, S.: The relationship between breaking the diffie-hellman protocol and computing discrete logarithms. SIAM J. Comput. 28(5), 1689–1721 (1999)

[MW00] Maurer, U.M., Wolf, S.: The diffie-hellman protocol. Des. Codes Cryptography 19(2/3), 147–171 (2000)

[OO91] Ohta, K., Okamoto, T.: A digital multisignature scheme based on the fiat-shamir scheme. In: Matsumoto, T., Imai, H., Rivest, R.L. (eds.) ASIACRYPT 1991. LNCS, vol. 739, pp. 139–148. Springer, Heidelberg (1993)

[OO99] Ohta, K., Okamoto, T.: Multisignature schemes secure against active insider attacks. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E82-A(1), 21–31 (1999)

[PKC00] PKCS#10. Certification request syntax standard. In: RSA Data Security, Inc. (2000)

[PS00] Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology 13(3), 361–396 (2000)

[RY07] Ristenpart, T., Yilek, S.: The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 228–245. Springer, Heidelberg (2007)

[Sho00] Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)

Using Normal Bases for Compact Hardware Implementations of the AES S-Box

Svetla Nikova[1], Vincent Rijmen[1,2], and Martin Schläffer[2]

[1] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
[2] Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
martin.schlaeffer@iaik.tugraz.at

Abstract. The substitution box (S-box) of the Advanced Encryption Standard (AES) is based on the multiplicative inversion $s(x) = x^{-1}$ in GF(256) and followed by an affine transformation in GF(2). The S-box is the most expansive building block of any hardware implementation of the AES, and the multiplicative inversion is the most costly step of the S-box transformation. There exist many publications about hardware implementations of the S-box and the smallest known implementations are based on normal bases. In this paper, we introduce a new method to implement the multiplicative inversion over GF(256) based on normal bases that have not been considered before in the context of AES implementations.

Keywords: AES, S-box, hardware implementation, normal basis.

1 Introduction

The first efficient hardware implementation of the multiplicative inversion in GF(256) has been proposed by Rijmen [9] and first imlemented by Rudra et al. [10] and Wolkerstorfer et al. [13]. They decompose the elements of GF(256) into polynomials of degree 2 over the subfield GF(16). In the next step, the elements of GF(16) are further decomposed into polynomials of degree 4 over GF(2). The resulting operations in GF(2) work on bit-level and can be implemented in hardware using simple gates.

Satoh et al. [11] and further Mentens et al. [6] use the different *tower field decomposition* in their implementation. They first start by decomposing the elements of GF(256) into polynomials over GF(16) as well. But then the elements of the field GF(16) are further decomposed into polynomials over the subfield GF(4) before implementing the final operations in GF(2). In all these approaches the field elements are represented by using polynomial bases. In contrast, Canright has been able to further reduce the size of the S-box computation by using normal bases at all levels of the tower field decomposition [3,2].

In this paper, we propose a new way to implement the inversion of the AES S-box. We use normal bases as in the approach of Canright but do not decompose

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 236–245, 2008.

the elements of GF(16) into elements of GF(4). Table 1 illustrates the relation between the four mentioned approaches. We show that our implementation of the AES S-box can be at least as compact as the one proposed by Canright when counting the required number of gates. Which of the approaches is the best depends not only on the used hardware technology, but also on the application of the circuit and the resulting hardware criteria like size, timing, or power consumption [12]. Therefore, we cannot make an unambiguous ranking of the different implementations, but we are convinced that our alternative has its merits, already simply because it increases the available options of a hardware designer.

Table 1. Four approaches to decompose elements of GF(256) into elements of smaller subfields

	Decomposition of field elements	
	GF(256) → GF(16) → GF(2)	GF(256) → GF(16) → GF(4) → GF(2)
Polynomial bases	Rijmen [9], Rudra et al. [10], Wolkerstorfer et al. [13]	Satoh et al. [11], Mentens el al. [6]
Normal bases	this paper	Canright [3,2]

The paper is organized as follows. In Section 2 we briefly recall some properties of normal bases which are relevant for the implementation of the AES S-box. Subsequently, in Section 3 we study extensively the different normal bases of GF(16) over GF(2) and discuss the impact of the choice of basis on the structure and complexity of the multiplicative inversion over GF(16). Additional hardware related considerations are made in Section 4 and we conclude in Section 5

2 Normal Bases

In this section we briefly summarize some properties of normal bases over finite fields [5]. The finite field $\mathrm{GF}(2^{mp})$ is isomorphic to a p-dimensional vector space over $\mathrm{GF}(2^m)$. This implies that it is possible to construct a *basis* for $\mathrm{GF}(2^{mp})$. A basis consists of p elements $\beta_0, \beta_2, \ldots, \beta_{p-1} \in \mathrm{GF}(2^{mp})$ such that all elements of $\mathrm{GF}(2^{mp})$ can be written as a linear combination of the elements β_j, with all coefficients elements of $\mathrm{GF}(2^m)$. As in all vector spaces, there are many different choices possible for the basis, and the choice of basis may influences the complexity to describe transformations on the vector space.

2.1 Construction

A *normal basis* is constructed by choosing an element $\theta \in \mathrm{GF}(2^{mp})$ and setting $\beta_j = \theta^{2^{mj}}$. Not all elements of $\mathrm{GF}(2^{mp})$ result in a basis, but there exist always some suitable elements. Let now

$$x = \sum_{j=0}^{p-1} c_j \theta^{2^{mj}}, \quad c_j \in \mathrm{GF}(2^m).$$

We raise both sides to the power 2^m. This operation is linear over $\mathrm{GF}(2^{mp})$ and corresponds to the identity transformation for all elements in $\mathrm{GF}(2^m)$. Then we obtain

$$x^{2^m} = \sum_{j=0}^{p-1} (c_j)^{2^m} (\theta^{2^{mj}})^{2^m} = \sum_{j=0}^{p-1} c_j \theta^{2^{m(j+1)}} = \sum_{j=0}^{p-1} c_{j-1} \theta^{2^{mj}}.$$

In words, this corresponds to the following property.

Property 1 ([5]). If the elements of the finite field $\mathrm{GF}(2^{mp})$ are represented by p-dimensional vectors over $\mathrm{GF}(2^m)$ using a normal basis, then raising an element to the power 2^m corresponds to rotating the coordinates of the element by one position.

Let now $m = 1$ and consider the inversion map used in the AES S-box. We denote this map by s. Then we have:

$$\begin{aligned} \mathrm{s}(0) &= 0, \\ \mathrm{s}(x) &= x^{-1}, \quad x \neq 0. \end{aligned}$$

Equivalently, we can write: $\mathrm{s}(x) = x^{254}$. Clearly, $\mathrm{s}(x^2) = x^{508} = (\mathrm{s}(x))^2$. Hence we obtain the following property.

Property 2. If the elements of the finite field $\mathrm{GF}(2^{mp})$ are represented in a normal basis, then the inversion map $\mathrm{s}(x)$ is rotation invariant:

$$\mathrm{rot}(\mathrm{s}(x)) = \mathrm{s}(\mathrm{rot}(x)).$$

For the remainder of this section, we set $p = 2$ and let v be an element of $\mathrm{GF}(2^{2m})$ such that $v + v^\ell = 1$ with $\ell = 2^m$. Then $\{v, v^\ell\}$ is a normal basis of $\mathrm{GF}(2^{2m})$ and the coordinate vectors consist of two elements of $\mathrm{GF}(2^m)$. We denote by g the trace of v over $\mathrm{GF}(2^m)$:

$$g = v^2 + v, \tag{1}$$

and $g \in \mathrm{GF}(2^m)$. We show now that the use of a normal basis leads to simple formulas for products and inverses of elements.

2.2 Multiplication

Let (a, b) and (c, d) be the coordinates of two elements of $\mathrm{GF}(2^{2m})$. The coordinates of the product are given by the following formula:

$$\begin{aligned} (e, f) = (a, b) \times (c, d) &\Leftrightarrow ev + fv^\ell = acv^2 + (ad + bc)v^{\ell+1} + bdv^{2\ell} \\ &\Leftrightarrow \begin{cases} e = (a + b)(c + d)g + ac \\ f = (a + b)(c + d)g + bd \end{cases} \end{aligned}$$

Here the element g is defined by (1).

2.3 Inversion

Let (a, b) be the coordinates of an element of GF(2^{2m}). The coordinates of the inverse element are given by the following formula:

$$\begin{aligned}(c,d) = (a,b)^{-1} &\Leftrightarrow 1 = (av + bv^{\ell})(cv + dv^{\ell}) \\ &\Leftrightarrow 1 = acv^2 + (ad + bc)(v^2 + v) + bd(v^2 + 1) \\ &\Leftrightarrow \begin{cases} c = ((a+b)^2 g + ab)^{-1} b \\ d = ((a+b)^2 g + ab)^{-1} a \end{cases} \end{aligned} \tag{2}$$

Again the element g is defined by (1).

3 Implementing the AES Inverse Using Normal Bases

Canright has investigated all 432 possible tower field decompositions using polynomial and normal bases in his work [3,2]. He has obtained the smallest hardware implementation of the S-box by using normal bases at all three levels. However, he did not consider the decomposition of the field GF(16) into the subfield GF(2) using normal bases. In this section we derive this decomposition and present formulas for the inversion in GF(16) using the additional normal bases.

3.1 Inversion in GF(256)

Assume a normal basis $\{v, v^{16}\}$. Equation (2) suggests the following 3-stage approach to implement the inverse [2,3].

Step 1: Compute the product ab and add the result to $(a + b)^2 g$. Computing the product is a nonlinear operation and the remaining operations are linear.
Step 2: Compute the multiplicative inverse over GF(16) of the result of Step 1.
Step 3: Multiply the result of Step 2 once with a, and once with b.

Note that Step 3 uses twice the same hardware. Hence, hardware can be saved by splitting Step 3 up into two steps using the same circuit. The computation of the multiplicative inverse over GF(16) can be computed by applying recursion, i.e. describing GF(16) as an extension field of GF(4) using the *tower field approach*. We have opted for a direct computation of the inverse over GF(16), as explained in the next section.

3.2 Normal Bases in GF(16)

The field GF(16) can be described directly as an extension field of GF(2). This approach gives a 4-dimensional basis with coordinate elements from GF(2), i.e. $m = 1$ and $p = 4$. The field GF(16) counts exactly 8 solutions to the following equation:

$$\theta + \theta^2 + \theta^4 + \theta^8 = 1. \tag{3}$$

These 8 elements define 8 possible normal bases: $\{\theta, \theta^2, \theta^4, \theta^8\}$. Note that if θ_1 satisfies (3), then so do θ_1^2, θ_1^4 and θ_1^8, i.e. they define rotated versions of the same 4 base vectors. Thus the number of the normal bases reduces to two, as noted in [7].

It is called an *optimal normal base (ONB)* [7], because the complexity of the multiplication formula in this basis is minimal, i.e. equal to $2n - 1 = 7$ [4]. In the non-optimal normal basis (NB), the complexity of the multiplication formula equals 9 [8]. In those two cases multiplying $x = (x_0, x_1, x_2, x_3)$ with $y = (y_0, y_1, y_2, y_3)$ results in $z = (z_0, z_1, z_2, z_3)$, where

$$\begin{aligned} &\text{NB:} \quad z_3 = x_2y_3 + x_3y_2 + x_1y_3 + x_3y_1 + x_3y_0 + x_0y_3 + x_2y_2 + x_0y_1 + x_1y_0. \\ &\text{ONB:} \ z_3 = x_3y_1 + x_0y_1 + x_0y_2 + x_1y_3 + x_1y_0 + x_2y_0 + x_2y_2. \end{aligned} \tag{4}$$

As noted by Paar [8] the multiplication in any normal basis is rotation symmetric, so the rest of the output bits z_0, z_1 and z_2 can be computed by rotating the input bits. Note that the multiplicative group of $GF(16)$ has order 15 and we know from group theory that this group is the direct product of two cyclic groups of order 3 and order 5. Remember that the intersection of these two subgroups is trivial.

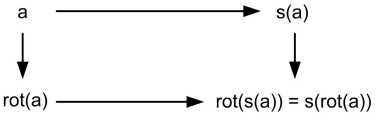
3.3 Inversion in GF(16)

We now study in detail how the choice of θ influences the complexity of the map $\mathrm{s}(x)$.

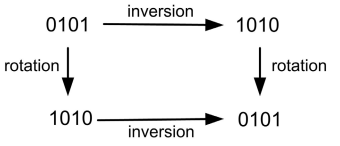
1. Equation (3) can be rewritten as

   $$(\theta + \theta^4)^2 + (\theta + \theta^4) = 1.$$

   It follows that for all θ satisfying (3), the elements $\theta + \theta^4$ and $\theta^2 + \theta^8$ are the roots of the polynomial $x^2 + x + 1$. The roots of this polynomial are exactly the two elements of order 3 in GF(16).
2. In a normal base, the zero element of GF(16) always has coordinates 0000, while the unit element always has 1111. Hence we always have $\mathrm{s}(0000) = (0000)$ and $\mathrm{s}(1111) = (1111)$.
3. It follows from Property 2 that x and $\mathrm{s}(x)$ must have coordinates with the same rotational symmetry. Hence, the inverse map must map elements with rotational symmetry to elements with the same symmetry. Hence we have the following commutative diagram:



   This implies that $\mathrm{s}(\{0101, 1010\}) = \{0101, 1010\}$. The inversion map has only two fixed points: the zero element and the unit element and it follows



that s(0101) = 1010 and s(1010) = 0101: We can conclude that 0101 and 1010 are the two elements of order 2. Together with the unit element, they form the cyclic subgroup of order 3 of the multiplicative group of $GF(16)$.

4. The remaining 12 elements of GF(16) can be partitioned into 3 sets with 4 elements each. The index of S_i denotes the number of ones in each representation:

$$S_1 = \{0001, 0010, 0100, 1000\}$$
$$S_2 = \{0011, 0110, 1100, 1001\}$$
$$S_3 = \{0111, 1110, 1101, 1011\}$$

Due to the rotational symmetry of s, this partitioning is consistent with s. If one element of S_i is mapped to an element of S_j, then all elements of S_i are mapped to elements of S_j, and since s is an involution, this also implies that all elements of S_j are mapped to elements of S_i.

5. Assume now that $s(S_i) = S_i$. Then the rotational symmetry of s implies:

$$\exists r, \forall v \in S_i : s(v) = \mathrm{rot}_r(v).$$

Since s is an involution we have

$$\mathrm{rot}_r(\mathrm{rot}_r(v)) = \mathrm{rot}_{2r}(v) \equiv v.$$

This implies that $2r = 0 \pmod 4$. Since we cannot have new fixed points for s, $r \neq 0$ we get $r = 2$. We require that $s(S_2) \neq S_2$ because otherwise, we would get s(0011) = (1100), and the corresponding element satisfies $x^{-1} + x = 1$. Since this would mean that it is a root of the polynomial $x^2 + x + 1$ we get a contradiction because the roots of this polynomial have coordinates (0101) and (1010). Hence, there are only two possibilities left: either $s(S_1) = S_1$ and $s(S_2) = S_3$, or $s(S_1) = S_2$ and $s(S_3) = S_3$. Further analysis shows that

$$s(S_1) = S_1 \Leftrightarrow s(0001) = (0100),$$
$$s(S_3) = S_3 \Leftrightarrow s(0111) = (1101),$$

because other choices lead to violations of Property 2. For each of these possibilities, there are 4 possibilities left for $s(0, 0, 1, 1)$ and this choice determines completely the action of s on the coordinate vectors. Table 2 lists the 8 remaining candidates for the inversion map in GF(16). The cases A-D correspond to the 4 choices for $S_1 = s(S_2)$ and the cases E-H to the 4 choices for $S_3 = s(S_2)$.

Table 2. 8 candidates for the inversion map in GF(16). The first 4 cases have $s(S_2) = S_1$, the last 4 have $s(S_2) = S_3$.

x	s(x)							
	A	B	C	D	E	F	G	H
0000	0000	0000	0000	0000	0000	0000	0000	0000
1111	1111	1111	1111	1111	1111	1111	1111	1111
0101	1010	1010	1010	1010	1010	1010	1010	1010
1010	0101	0101	0101	0101	0101	0101	0101	0101
0001	0011	0110	1100	1001	0100	0100	0100	0100
0010	0110	1100	1001	0011	1000	1000	1000	1000
0100	1100	1001	0011	0110	0001	0001	0001	0001
1000	1001	0011	0110	1100	0010	0010	0010	0010
0011	0001	1000	0100	0010	0111	1011	1101	1110
0110	0010	0001	1000	0100	1110	0111	1011	1101
1100	0100	0010	0001	1000	1101	1110	0111	1011
1001	1000	0100	0010	0001	1011	1101	1110	0111
0111	1101	1101	1101	1101	0011	0110	1100	1001
1011	1110	1110	1110	1110	1001	0011	0110	1100
1101	0111	0111	0111	0111	1100	1001	0011	0110
1110	1011	1011	1011	1011	0110	1100	1001	0011

The two choices $S_1 = s(S_2)$ and $S_3 = s(S_2)$ can be restated as $S_3 = s(S_3)$ and $S_1 = s(S_1)$. Hence, the choice is which elements will have order 5 and will form together with the unit element the cyclic subgroup of order 5 of the multiplicative group of $GF(16)$.

Recall that the two cyclic groups of order 3 and order 5 build up the multiplicative group of $GF(16)$. Thus the fact that the cyclic group of order 3 is fixed and for the cyclic group of order 5 we have 2 choices implies that we have two choices for the multiplicative group of $GF(16)$ this naturally corresponds to the fact that we have only two normal bases in $GF(16)$.

6. Finally, we use the fact that s must satisfy

$$s(xy) = s(x)s(y), \forall x, y$$

to eliminate all but two of the candidate maps. We obtain that only case A and case H are inversion maps, corresponding to the optimal, respectively the normal base, mentioned before.

Table 3 gives the truth tables for the Boolean functions that computes the rightmost bit f_0 of s(x) in the two cases:

$$(f_3, f_2, f_1, f_0) = s(x_3, x_2, x_1, x_0)$$

Due to the rotational symmetry of s, the other output bits f_3, f_2 and f_1 can be computed by rotating the input bits. The Algebraic Normal Form (ANF) of each output bit f_0 is given by:

$$\begin{array}{ll} \text{NB:} & f_0 = x_0 + x_3 + x_0x_1 + x_1x_3 \qquad\quad + x_0x_1x_2 + x_0x_1x_3 + x_1x_2x_3 \\ \text{ONB:} & f_0 = x_1 \qquad + x_0x_3 + x_0x_2 + x_1x_3 + x_0x_1x_2 + x_0x_1x_3 + x_1x_2x_3 \end{array} \quad (5)$$

Table 3. The truth tables for the Boolean functions computing the rightmost bit of $s(x)$ in the two cases

x	A (NB)	H (ONB)
0000	0	0
0001	1	0
0010	0	0
0011	1	0
0100	0	1
0101	0	0
0110	0	1
0111	1	1
1000	1	0
1001	0	1
1010	1	1
1011	0	0
1100	0	1
1101	1	0
1110	1	1
1111	1	1

In the next section we show how the ANF can be simplified to reduce the number of operations and the resulting hardware size.

4 Hardware Considerations

In this section we optimize the hardware implementation of the different normal bases regarding their size. We present the size of the inversion in GF(16) using the two normal bases and compare our results with the results of Canright.

4.1 Optimizing the Implementation

In order to achieve minimal hardware implementations, the 4 formulae for the 4 output bits of the inversion in GF(16) need to be further optimized. We compute the four output bits in parallel and share intermediate terms between different functions. Hence, formulae which allow to share many terms, have an advantage.

Secondly, '+' operations (XOR) are usually very expensive in hardware implementations. For instance using the AMS $0.35\mu m$ technology [1], an XOR gate costs 2.33 times more than a NAND gate. Hence, the final implementation formulae should contain as little '+' operations as possible. For instance, the inversion formulae using the optimal normal basis (5) can be rewritten as:

$$\begin{aligned}
f_0 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_0, \overline{x_2})), \mathrm{NAND}(x_3, \overline{x_1}),\\
&\qquad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_0}, x_3), x_1), \overline{x_2})), \mathrm{NAND}(x_1, \overline{x_3}))\\
f_1 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_3, \overline{x_1}), \mathrm{NAND}(x_2, \overline{x_0})),\\
&\qquad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_3}, x_2), x_0), \overline{x_1})), \mathrm{NAND}(x_0, \overline{x_2}))\\
f_2 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_2, \overline{x_0})), \mathrm{NAND}(x_1, \overline{x_3}),\\
&\qquad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_2}, x_1), x_3), \overline{x_0})), \mathrm{NAND}(x_3, \overline{x_1}))\\
f_3 &= \mathrm{NAND}(\mathrm{NOR}(\mathrm{NOR}(\mathrm{NAND}(x_1, \overline{x_3})), \mathrm{NAND}(x_0, \overline{x_2}),\\
&\qquad \mathrm{NOR}(\mathrm{NOR}(\mathrm{NOR}(\overline{x_1}, x_0), x_2), \overline{x_3})), \mathrm{NAND}(x_2, \overline{x_0}))
\end{aligned} \tag{6}$$

These formulae use 16 NAND gates, 20 NOR gates, 20 NOT gates. When implementing the 4 output bits in parallel, the inverters $\overline{x_i}$ and the common terms NAND$(x_i, \overline{x_{i+2}})$ can be shared between different output bits. Therefore, these 4 functions can finally be implemented using 20 NOR gates, 8 NAND gates and 4 NOT gates.

4.2 Counting Gate Equivalents

We refer to the size of the NAND gate by one gate equivalent (GE). In the AMS $0.35\mu m$ standard cell library [1] an XNOR gate corresponds to 2GE, an XOR gate to 2.33GE, an inverter (INV) to 0.65GE and a NOR gate to 1GE. These values give a total of 30.7GE for our best GF(16) inversion. This compares favorably to the best GF(16) inversion circuit reported by Canright, which uses 9 XOR and 10 NAND gates, corresponding to 31.0GE [2]. Referring to personal communication with Satoh, Canright equates XOR and XNOR gates to 1.75GE, and inverters to 0.75GE. Using these values, our best GF(16) inversion costs 31.0GE, but the cost of the best Canright implementation is reduced to 25.8GE.

However, standard cell libraries provide additional operations other than the basic Boolean operations INV, NAND, NOR, XNOR and XOR. For example, there are extended operations with more than one input which can be used to improve the 3-input NOR terms of (6). Additionally, the AMS $0.35\mu m$ library provides specialized standard cells like AND-OR-INVERT (AOI) which compute $Q = A.B+C+1$, or OR-AND-INVERT (OAI) which compute $Q = (A+B).C+1$. Both can be implemented using only $1.33GE$ [1] and have far less size than the XOR or XNOR gates. Using these additional cells, we have been able to improve the GF(16) inversion of Canright to $26.7GE$ and the inversion based on the optimal normal base formulae to 22.7GE. Table 4 gives an overview of these results.

Table 4. Equivalent gate costs for the implementation of several GF(16) inversions

Design	Canright's GE [3]	basic standard cells [1]	all standard cells [1]
Canright	25.8	31.0	26.7
NB	34.3	34.0	24.7
ONB	31.0	30.7	22.7

5 Conclusion

In this paper, we discussed alternative constructions for the AES S-box using normal bases for GF(16) over GF(2). We worked out example implementations showing that our normal bases can compete with the results of Canright. Of course, the final size of the S-box depends on the size of the multiplication in GF(16) and on the complexity of the basis transformations as well. As shown in Section 4.2, also the target technology influences the final count on the implementation cost. Therefore, our normal bases should at least be considered when designing small AES S-box implementations.

We did not check all possible cases, since the result can only be a specialized implementation for a single target technology. Further, the best basis does not only depend on the hardware size but on other optimization constraints such as low power, timing and throughput as well. However, using our normal bases new promising alternatives for hardware designers of compact AES S-boxes are available.

References

1. Austria Microsystems. Standard Cell Library 0.35μm CMOS (C35), http://asic.austriamicrosystems.com/databooks/c35/databook_c35_33
2. Canright, D.: A very compact Rijndael S-box (May 2005), http://web.nps.navy.mil/~dcanrig/pub.
3. Canright, D.: A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
4. Certicom. F_{2^4} with Optimal Normal Basis Representation, http://www.certicom.com/index.php?action=ecc_tutorial,math9_1
5. Lidl, R., Niederreiter, H.: Introduction to Finite Fields and their Applications. Cambridge University Press, New York (1986)
6. Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 323–333. Springer, Heidelberg (2005)
7. Mullin, R.C., Onyszchuk, I.M., Vanstone, S.A., Wilson, R.M.: Optimal Normal Bases in $GF(p^n)$. Discrete Appl. Math. 22, 149–161 (1989)
8. Paar, C.: Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. PhD thesis, Institute for Experimental Mathematics, University of Essen (1994)
9. Rijmen, V.: Efficient Implementation of the Rijndael S-box (2000), www.iaik.tugraz.at/RESEARCH/krypto/AES/old/~rijmen/rijndael/sbox.pdf
10. Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R., Rohatgi, P.: Efficient rijndael encryption implementation with composite field arithmetic. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 171–184. Springer, Heidelberg (2001)
11. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
12. Tillich, S., Feldhofer, M., Großschädl, J., Popp, T.: Area, Delay, and Power Characteristics of Standard-Cell Implementations of the AES S-Box. Journal of Signal Processing Systems 50(2), 251–261 (2008)
13. Wolkerstorfer, J., Oswald, E., Lamberger, M.: An ASIC Implementation of the AES SBoxes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 67–78. Springer, Heidelberg (2002)

A New Analysis of the McEliece Cryptosystem Based on QC-LDPC Codes

Marco Baldi[1], Marco Bodrato[2], and Franco Chiaraluce[1]

[1] DEIT, Università Politecnica delle Marche
Ancona, Italy
{m.baldi,f.chiaraluce}@univpm.it
[2] Centro Vito Volterra, Università di Roma Tor Vergata
Roma, Italy
bodrato@mail.dm.unipi.it

Abstract. We improve our proposal of a new variant of the McEliece cryptosystem based on QC-LDPC codes. The original McEliece cryptosystem, based on Goppa codes, is still unbroken up to now, but has two major drawbacks: long key and low transmission rate. Our variant is based on QC-LDPC codes and is able to overcome such drawbacks, while avoiding the known attacks. Recently, however, a new attack has been discovered that can recover the private key with limited complexity. We show that such attack can be avoided by changing the form of some constituent matrices, without altering the remaining system parameters. We also propose another variant that exhibits an overall increased security level. We analyze the complexity of the encryption and decryption stages by adopting efficient algorithms for processing large circulant matrices. The Toom-Cook algorithm and the short Winograd convolution are considered, that give a significant speed-up in the cryptosystem operations.

Keywords: McEliece cryptosystem, QC-LDPC codes, Cryptanalysis, Toom-Cook, Winograd.

1 Introduction

The McEliece cryptosystem is a public-key cryptosystem based on algebraic coding theory [1] that revealed to have a very high security level. It adopts a generator matrix as the private key and one transformation of it as the public key, while its security lies in the difficulty of decoding a large linear code with no visible structure, that is known to be an NP complete problem [2].

The original McEliece cryptosystem is still unbroken, in the sense that a total break attack has never been found, and even local deduction attacks remain quite intractable in practice [3,4]. Moreover, the system is two or three orders of magnitude faster than competing solutions, like RSA, that is among the most popular public key algorithms currently in use. Despite this, the McEliece cryptosystem has been rarely considered in practical applications; this is due to the fact it exhibits two major drawbacks: i) large size of the public key and ii) low transmission rate (that is about 0.5).

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 246–262, 2008.

In his original formulation, McEliece used Goppa codes of length $n = 1024$, dimension $k = 524$, and minimum distance $d_{\min}$ of at least 101, that are able to correct $t = 50$ errors. Several attempts have been made, later on, for overcoming the drawbacks of the original system and/or further reducing the complexity, but the adoption of alternative families of codes has not been possible without compromising the system security. Generalized Reed-Solomon (GRS) codes, in particular, were initially considered for an important variant of the McEliece cryptosystem, proposed by Niederreiter [5], but they revealed to be unsecure. On the other hand, when employing Goppa codes, the Niederreiter cryptosystem shows some advantages: it has equivalent security to that of the McEliece system, for codes with the same parameters [6], but it requires a key size more than halved (when considering the values reported above), a transmission rate slightly increased, and the possibility to use a public key in systematic form. For these reasons, though renouncing to use GRS codes, the Niederreiter system is considered a good alternative to the McEliece system.

A clever technique for increasing the transmission rate has been proposed by Riek [7], and consists in mapping additional information bits onto the intentional error vector. This approach could increase significantly the transmission rate, but requires the introduction of an additional encoding and decoding stage to implement a positional code on error vectors.

Low-Density Parity-Check (LDPC) codes represent the state of the art in forward error correction techniques, and permit to approach the theoretical Shannon limit [8], while ensuring limited complexity. Quasi-cyclic (QC) LDPC codes are a particular class of LDPC codes, able to join low complexity encoding of QC codes with high-performing and low-complexity decoding techniques based on the belief propagation principle. Several classes of QC-LDPC codes have been proposed up to now, all having in common the parity-check matrix structure, that is formed by sparse circulant blocks.

In a recent work, we have proposed to adopt a particular family of QC-LDPC codes in the McEliece cryptosystem to reduce the key size and increase the transmission rate [9]. We have shown such variant is able to counter all the general attacks, and even new attacks that can compromise the security of previous LDPC-based versions of the cryptosystem, like that proposed in [10].

Very recently, however, Otmani, Tillich and Dallot developed a new attack that, exploiting a flaw in the transformation from the private key to the public key, is able to recover the secret key with very high probability [11]. They presented three attack strategies, that will be denoted as OTD1, OTD2 and OTD3 in the following. In the same work, the authors also proved that a previous proposal for the adoption of quasi-cyclic (but not LDPC) codes for shortening the public key of the McEliece cryptosystem [12] is not secure.

In this paper, we shortly describe the three OTD attacks, and analyze the flaw in the private-public key map that originates them. We propose a first variant of the cryptosystem that is able to counter such attacks by adopting a different form for its constituent matrices, without altering other parameters.

Furthermore, we present a second variant of the cryptosystem that provides overall increased security.

In addition, we study the application, in this new version of the cryptosystem, of efficient algorithms for computation on large circulant matrices, that permit to reduce its encryption and decryption complexity. The main question concerns encoding, that, in the case of QC codes, can be implemented through a barrel shift-register with hardware complexity that increases linearly with the code length. However, the total number of operations can still be high, thus reflecting in latency issues. A common solution to this problem consists in searching for sparse generator matrices [13,14]. Though this is possible for suitably designed codes, such approach is unsuitable for codes randomly designed for the use in cryptographic systems. In this case, however, efficient computation algorithms can limit the number of operations. In this paper, we consider two possible choices of efficient multiplication algorithms, namely, the Toom-Cook algorithm and the short Winograd convolution, that actually yield reduced complexity.

2 Notation

Let $\mathbb{F} = GF(m)$ be the Galois field of order m, with m a prime power.

A $p \times p$ Toeplitz matrix $\mathbf{A}$ over $\mathbb{F}$ is defined as follows:

$$\mathbf{A} = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{p-1} \\ a_{-1} & a_0 & a_1 & \cdots & a_{p-2} \\ a_{-2} & a_{-1} & a_0 & \cdots & a_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1-p} & a_{2-p} & a_{3-p} & \cdots & a_0 \end{bmatrix} . \tag{1}$$

It is called circulant if $\forall i, a_i = a_{i-p}$. In this work we restrict our analysis to binary circulant matrices, that is, we consider $m = 2$.

There is a natural isomorphism between the algebra of $p \times p$ circulant matrices with entries in the field $\mathbb{F}$ and the ring of polynomials $\mathbb{F}[x]/(x^p + 1)$. If we denote by $\mathbf{X}$ the matrix with entries:

$$X_{i,j} = \begin{cases} 1 \text{ if } j - i \equiv 1 \pmod{p} \\ 0 \text{ if } j - i \not\equiv 1 \pmod{p} \end{cases} , \tag{2}$$

the isomorphism maps the matrix $\mathbf{X}$ to the monomial x and the generic circulant matrix $\sum_{i=0}^{p-1} \alpha_i \mathbf{X}^i$ to the polynomial $\sum_{i=0}^{p-1} \alpha_i x^i \in \mathbb{F}[x]/(x^p + 1)$. This isomorphism can be extended to matrices with circulant blocks, as will be shown in the next section.

3 Improved McEliece Cryptosystem Based on QC-LDPC Codes

In order to hide the secret code's structure, we have recently proposed a variant of the McEliece cryptosystem working as follows. Bob, in order to receive encrypted

messages, randomly chooses a code in a family of (n_0, d_v, p) QC-LDPC codes based on Random Difference Families [9], by selecting its parity-check matrix $\mathbf{H}$, and produces a generator matrix $\mathbf{G}$ in reduced echelon form. Matrix $\mathbf{H}$ is formed by a row $\{\mathbf{H}_0, \ldots, \mathbf{H}_{n_0-1}\}$ of n_0 binary circulant blocks with size p and row/column weight d_v, and it is part of Bob's private key. Matrix $\mathbf{G}$, instead, is formed by a $k \times k$ identity matrix $\mathbf{I}$ (with $k = k_0 \cdot p$ and $k_0 = n_0 - 1$), followed by a column of k_0 binary circulant blocks with size p. If we suppose $\mathbf{H}_{n_0-1}$ to be non-singular, $\mathbf{G}$ can be obtained as follows:

$$\mathbf{G} = \begin{bmatrix} & \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_0\right)^T \\ & \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_1\right)^T \\ \mathbf{I} & \vdots \\ & \left(\mathbf{H}_{n_0-1}^{-1} \cdot \mathbf{H}_{n_0-2}\right)^T \end{bmatrix} . \tag{3}$$

The remaining part of Bob's private key is formed by two other matrices: a $k \times k$ non-singular matrix $\mathbf{S}$ and a sparse $n \times n$ non-singular matrix $\mathbf{Q}$. $\mathbf{S}$ and $\mathbf{Q}$ are regular matrices, formed by $k_0 \times k_0$ and $n_0 \times n_0$ blocks of $p \times p$ binary circulants, respectively. $\mathbf{Q}$ has row/column weight m.

By exploiting the isomorphism introduced in Section 2, the generator matrix $\mathbf{G}$ can be seen as a $k_0 \times n_0$ matrix with entries in the ring of polynomials $\mathbb{R} = GF(2)[x]/(x^p + 1)$; $\mathbf{S}$ and $\mathbf{Q}$ are invertible matrices on the same ring with size $k_0 \times k_0$ and $n_0 \times n_0$, respectively.

Then, Bob computes the public key as follows:

$$\mathbf{G}' = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{Q}^{-1} . \tag{4}$$

It should be noted that $\mathbf{G}'$ preserves the quasi-cyclic structure of $\mathbf{G}$, due to the block circulant form of $\mathbf{S}$ and $\mathbf{Q}$.

$\mathbf{G}'$ is made available in a public directory. Alice, who wants to send an encrypted message to Bob, extracts $\mathbf{G}'$ from the public directory and divides her message into k-bit blocks. If $\mathbf{u}$ is one of these blocks, Alice obtains the encrypted version as follows:

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e} = \mathbf{c} + \mathbf{e} .$$

In this expression, $\mathbf{e}$ is a vector of intentional errors randomly generated, with length n and weight t'.

When Bob receives the encrypted message $\mathbf{x}$, he first computes:

$$\mathbf{x}' = \mathbf{x} \cdot \mathbf{Q} = \mathbf{u} \cdot \mathbf{S}^{-1} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{Q} . \tag{5}$$

Vector $\mathbf{x}'$ is a codeword of the LDPC code chosen by Bob (corresponding to the information vector $\mathbf{u}' = \mathbf{u} \cdot \mathbf{S}^{-1}$), affected by the error vector $\mathbf{e} \cdot \mathbf{Q}$, whose maximum weight is $t = t' \cdot m$. If t' and m are suitably chosen, Bob is able to correct all the errors with very high probability, by means of LDPC decoding, thus recovering $\mathbf{u}'$, and then $\mathbf{u}$ through a post-multiplication by $\mathbf{S}$.

In the version of QC-LDPC based McEliece cryptosystem proposed in [9], we fixed $n_0 = 4$, $d_v = 13$, $p = 4032$, $m = 7$ and $t' = 27$. Both $\mathbf{S}$ and $\mathbf{Q}$ were chosen

sparse, and their non-null circulant blocks had row/column weight equal to m. In particular, the following block-diagonal form for $\mathbf{Q}$ was adopted:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{n_0-1} \end{bmatrix} \tag{6}$$

that, jointly with its low density, gives raise to the flaw exploited by the OTD attack, shortly described in the following subsection. In addition, as pointed out in [11], matrix $\mathbf{S}$ should contain some null blocks in order to be non-singular.

3.1 The OTD Attack

The adoption of a sparse $\mathbf{S}$ and a sparse block-diagonal $\mathbf{Q}$ implies that, by simply selecting the first k columns of the public key $\mathbf{G}'$, obtained through the transformation (4) with $\mathbf{G}$ in the form (3), an eavesdropper can derive the following matrix:

$$\mathbf{G}'_{\leq k} = \mathbf{S}^{-1} \cdot \begin{bmatrix} \mathbf{Q}_0^{-1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1^{-1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{Q}_{n_0-2}^{-1} \end{bmatrix} . \tag{7}$$

By calculating the inverse of $\mathbf{G}'_{\leq k}$ and considering its circulant block at position (i, j), the eavesdropper can easily obtain $\mathbf{Q}_i\mathbf{S}_{i,j}$, being $\mathbf{S}_{i,j}$ the circulant block at position (i, j) in matrix $\mathbf{S}$. Because of the isomorphism, this matrix corresponds to the polynomial:

$$g_{i,j}(x) = q_i(x) \cdot s_{i,j}(x) \bmod (x^p + 1) \tag{8}$$

where polynomials $q_i(x)$ and $s_{i,j}(x)$, in turn, correspond to the blocks $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$, respectively.

Due to the fact that both $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$ are sparse (they have row/column weight m), the vector of coefficients of $g_{i,j}(x)$ is obtained as the cyclic convolution of two sparse vectors containing the coefficients of $q_i(x)$ and $s_{i,j}(x)$. For this reason, it is highly probable that $g_{i,j}(x)$ has exactly m^2 non-null coefficients, and its support contains at least one shift $x^{l_a} \cdot q_i(x)$, $0 \leq l_a \leq p-1$ [11]. This is the initial point for three different attack strategies that, starting from the knowledge of $g_{i,j}(x)$, aim at revealing part of the secret key. They are briefly described next.

OTD1 Attack Strategy. The first attack strategy consists in enumerating all the m-tuples that belong to the support of $g_{i,j}(x)$. Each m-tuple is then validated through inversion of its corresponding polynomial and multiplication by $g_{i,j}(x)$. If the resulting polynomial has exactly m non-null coefficients, the considered m-tuple corresponds to a shifted version of $q_i(x)$ with very high probability. For the specified numerical values, this attack requires a work factor of $2^{50.3}$ binary operations.

OTD2 Attack Strategy. The second attack strategy is based on the periodic autocorrelation of the coefficients vector of $g_{i,j}(x)$. In fact, it is highly probable that the Hadamard product of the polynomial $g_{i,j}(x)$ with a d-shifted version of itself, $g^d_{i,j}(x) * g_{i,j}(x)$, results in a shifted version of $q_i(x)$, for a specific value of d. For this reason, the eavesdropper can calculate all the possible $g^d_{i,j}(x) * g_{i,j}(x)$ and check whether the resulting polynomial has support with weight m. This attack requires a work factor of 2^{36} binary operations, that could be even further reduced by calculating the periodic autocorrelation of the coefficients of $g_{i,j}(x)$ (this can be made through efficient algorithms), in order to find the value of d.

OTD3 Attack Strategy. The third attack strategy consists in considering the i-th row of the inverse of $\mathbf{G}'_{\leq k}$, that is:

$$\mathbf{R}_i = [\mathbf{Q}_i\mathbf{S}_{i,0}|\mathbf{Q}_i\mathbf{S}_{i,1}|\dots|\mathbf{Q}_i\mathbf{S}_{i,n_0-2}] \tag{9}$$

and the linear code generated by

$$\mathbf{G}_{OTD3} = (\mathbf{Q}_i\mathbf{S}_{i,0})^{-1} \cdot \mathbf{R}_i = \left[\mathbf{I}|\mathbf{S}^{-1}_{i,0}\mathbf{S}_{i,1}|\dots|\mathbf{S}^{-1}_{i,0}\mathbf{S}_{i,n_0-2}\right]. \tag{10}$$

Such code admits an alternative generator matrix in the following form:

$$\mathbf{G}'_{OTD3} = \mathbf{S}_{i,0}\mathbf{G}_{OTD3} = [\mathbf{S}_{i,0}|\mathbf{S}_{i,1}|\dots|\mathbf{S}_{i,n_0-2}] \tag{11}$$

that coincides with a block row of matrix $\mathbf{S}$. Since matrix $\mathbf{S}$ has been chosen sparse, the code contains low weight codewords. Precisely, $\mathbf{G}'_{OTD3}$ has row weight equal to $m(n_0 - 1)$, that is very small compared to the code length.

Low weight codewords can be effectively searched through Stern's algorithm [15] and its variants [4]. Once having found matrix $\mathbf{S}$, a significant part of the secret key can be revealed by using (7). For the present choice of the system parameters, searching for low weight codewords in the code generated by $\mathbf{G}_{OTD3}$ would require, on average, 2^{32} binary operations.

3.2 First Variant of the Cryptosystem

The fundamental issue that validates the three OTD attack strategies relies in the fact that both $\mathbf{S}$ and $\mathbf{Q}$ are sparse and that matrix $\mathbf{Q}$ has block-diagonal form.

However, the three OTD attacks can be countered by adopting dense $\mathbf{S}$ matrices, without altering the remaining system parameters. For example, $\mathbf{S}$ could have row/column weight approximately equal to $k_0p/2$, with odd weight blocks along the main diagonal, and even weight blocks elsewhere, in order to assure non-singularity of $\mathbf{S}$, so that no further check is needed.

The adoption of dense $\mathbf{S}$ matrices prevents the eavesdropper from obtaining $\mathbf{Q}_i$ and $\mathbf{S}_{i,j}$, even knowing $\mathbf{Q}_i\mathbf{S}_{i,j}$. In this case, in fact, the probability that $g_{i,j}(x)$ has exactly m^2 non-null coefficients, and that its support contains that of at least one shift of $q_i(x)$ becomes extremely small. Furthermore, when $\mathbf{S}$ is dense, the code generated by $\mathbf{G}_{OTD3}$ does not contain any more low weight codewords, so all the three OTD attacks strategies are countered.

This modification has no effect on the number of errors to be corrected by the secret code, since the error spreading is only due to matrix $\mathbf{Q}$, that is kept sparse (with row/column weight m).

On the other hand, the choice of a dense $\mathbf{S}$ influences complexity of the decoding stage, that, however, can be reduced by resorting to efficient computation algorithms for circulant matrices. Complexity of the cryptosystem with dense $\mathbf{S}$ will be evaluated in Section 7.

As concerns matrix $\mathbf{Q}$, the OTD attacks demonstrate that the choice of the block-diagonal form is weak from the security viewpoint, so we avoid it in the revised versions of the cryptosystem. For example, an alternative choice would consist in obtaining $\mathbf{Q}$ from a matrix of $n_0 \times n_0 = 4 \times 4$ circulant blocks with weight 2, except those along the main diagonal, that have weight 1, and by permuting randomly its block rows and columns.

In this case, the inclusion of very low weight blocks in matrix $\mathbf{Q}$ could seem a flaw. However, the absence of the block-diagonal structure prevents from attacking each single block, and attacking a whole row or column would be too involved (it would require $p\binom{p}{2}^3 \approx 2^{81}$ attempts).

3.3 Second Variant of the Cryptosystem

In this second variant, we adopt the following set of parameters: $n_0 = 3$, $d_v = 13$ and $p = 8192$. The increased security level is achieved at the cost of a slightly decreased transmission rate, from 0.75 to 0.67, that, however, remains higher than in the original version. On the other hand, the reduction in the total number of circulant blocks permits to double their size without increasing the key length.

Both the private and the public codes, in this system, have dimension $k_0 p = 16384$ bits and length $n_0 p = 24576$. By means of numerical simulations, we have verified that such QC-LDPC codes are able to correct up to more than 470 errors per frame. For such reason, it has been possible to choose $t' = 40$ and $m = 11$ for this variant of the cryptosystem.

Matrix $\mathbf{Q}$ is formed by $n_0 \times n_0 = 3 \times 3$ circulant blocks with size p, and has row/column weight equal to $m = 11$. In this case, a possible choice consists in obtaining $\mathbf{Q}$ from a matrix of $n_0 \times n_0$ circulant blocks with weight 4, except those along the main diagonal, that have weight 3, and by permuting randomly its block rows and columns. In this case, attacking a whole row or column of $\mathbf{Q}$ would require $\binom{p}{4}^2\binom{p}{3} \approx 2^{131}$ attempts.

Matrix $\mathbf{S}$, instead, is formed by $k_0 \times k_0 = 2 \times 2$ circulant blocks with size p and it is dense, with row/column weight approximately equal to $k_0 p/2$. All its blocks have even row/column weight, except those along the main diagonal, that have odd weight, in order to allow non-singularity of the matrix.

3.4 Other Attacks

Having discussed above the OTD attack, in the following we list some of the other most important attacks with their corresponding work factor for the second cryptosystem variant, in order to assess its security. For a thorough description

of all the attack techniques already considered, we refer the interested reader to [16]. where it is also proved that they are avoided for the parameters chosen in the previous version of the cryptosystem (and, hence, in the first variant here proposed).

Brute force attacks. Brute force attacks could be tempted by enumerating all possible secret matrices $\mathbf{H}$. However, the number of equivalent QC-LDPC codes with the proposed choice of the system parameters is $> 2^{386}$. Even considering a single circulant block in $\mathbf{H}$, the number of possible choices would be $> 2^{122}$.

Information set decoding attacks. Information set decoding attacks could be tempted through two different strategies. The first one consists in Lee and Brickell's method [3], that, however, would require 2^{94} operations.

Alternatively, the vector of intentional errors $\mathbf{e}$ could be searched as the lowest weight codeword in the extended code generated by $\mathbf{G}'' = \begin{bmatrix} \mathbf{G}' \\ \mathbf{x} \end{bmatrix}$. With the new choice of the parameters, however, this search would be very involved. By using the Stern algorithm, for example, it would require more than 2^{82} operations. For the first variant, instead, a similar search would require 2^{71} operations.

At the current stage of cryptanalysis, these attacks achieve the minimum work factor, that can be hence considered as the security level of each cryptosystem variant.

Attacks to the dual code. When the sparse structure of the private $\mathbf{H}$ is not sufficiently hidden, the eavesdropper could recover it by searching for low weight codewords in the dual of the public code, that admits $\mathbf{G}'$ has parity-check matrix. In this version of the cryptosystem, however, the dual of the public code does not contain low weight codewords, due to the effect of matrix $\mathbf{Q}$ on the rows of the private matrix $\mathbf{H}$.

In the present case, matrix $\mathbf{Q}$ has row/column weight 11, so the dual of the public code has codewords with weight $\leq n_0 \cdot d_v \cdot m = 429$. Due to the fact that the rows of $\mathbf{H}$ are sparse, it is highly probable that the minimum weight codewords in the dual of the public code have weight close to $n_0 \cdot d_v \cdot m$. However, even a lower weight would suffice to avoid attacks to the dual code. For example, if we suppose the existence of $p = 8192$ codewords with weight 150 in the dual of the public code (that has length $n = 24576$ and dimension $p = 8192$), searching for one of them through the Stern algorithm would require more than 2^{92} operations.

4 Fast Computations with Circulant Matrices

By exploiting the isomorphism described in Section 2, between the algebra of $p \times p$ binary circulant matrices and the ring of polynomials $\mathbb{R} = GF(2)[x]/(x^p + 1)$, computing the determinant of $\mathbf{S}$ and $\mathbf{Q}$ to check their invertibility, and computing the $k_0 \times n_0$ public matrix $\mathbf{G}' = \mathbf{S}^{-1}\mathbf{G}\mathbf{Q}^{-1}$, require only a few tens of operations in the ring $\mathbb{R}$. Anyway, the key generation process is not the critical one for efficiency, so we will focus complexity analysis on the vector-matrix products used for encryption and decryption.

4.1 Vector-Matrix Product

The isomorphism extends also to vector-matrix product. Let us suppose to have a vector $u = (u_0, u_1, \ldots, u_{p-1})$, and a circulant $p \times p$ matrix $\mathbf{A}$ mapped to the polynomial $a(x) \in \mathbb{R}$ by the isomorphism. The product $\mathbf{w} = \mathbf{u} \cdot \mathbf{A} = (w_0, w_1, \ldots, w_{p-1})$ will then be the vector whose components satisfy the equation $\sum_{i=0}^{p-1} w_i x^i \equiv (\sum_{i=0}^{p-1} u_i x^i) \cdot a(x) \mod (x^p + 1)$. This vector-matrix product computation can be accelerated basically with two possible strategies: using fast polynomial multiplication algorithms based on evaluation-interpolation strategies, or using optimized vector-matrix product exploiting the Toeplitz structure.

To compare the methods, we will count the total number of bit operations needed. We will consider, for the naïve implementation, a number of operations given by the number of non-zero entries in the matrix. For the dense scenario we will consider that half of the entries are non-null. The starting value is hence $p^2/2$ operations. The two phases we will focus on are:

– The product $\mathbf{u} \cdot \mathbf{G}'$ used for encryption, where $\mathbf{u}$ is the message, seen as a vector of k_0 elements in $\mathbb{R}$, and $\mathbf{G}'$ is the public $k_0 \times n_0$ matrix with entries in $\mathbb{R}$. The cost with the naïve estimate is $p^2 k_0 n_0/2$ operations.

– The last step of decryption, that is, the product $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$, where $\mathbf{u}'$ is again a k_0 vector in $\mathbb{R}$, and $\mathbf{S}$ is a $k_0 \times k_0$ invertible matrix. The naïve cost for this step is $p^2 k_0^2/2$.

5 Fast Polynomial Product

All the algorithms for fast polynomial multiplication are based on the same scheme: evaluation, point-wise multiplication, interpolation. The first strategy of this kind was proposed by Karatsuba [17] and then generalized by Toom and Cook [18,19]. We will call both of them Toom-s, with s the splitting order [20].

Other asymptotically faster algorithms exist for $GF(2)$; the most interesting ones are due to Cantor and Schönhage [21,22]. Another approach is the use of segmentation, also known as Kronecker-Schönhage's trick, but the threshold between Toom-Cook and all these methods is far above 100 000 bits [23].

5.1 General Toom-Cook Approach

The Toom-s algorithm for polynomial product requires five steps:

– **Splitting:** The two operands are represented by two polynomials (f and g) with s coefficients.

– **Evaluation:** f and g are evaluated in $2s - 1$ points.

– **Point-wise multiplication:** Computed evaluations are multiplied to obtain evaluations of the product, for example $(f \cdot g)(0) = f(0) \cdot g(0)$.

– **Interpolation:** Once the values of the product $f \cdot g$ in $2s - 1$ points are known, the coefficients are obtained via interpolation.

– **Recomposition:** The coefficients of the result are combined to obtain the product of the original operands.

Starting from [24], where the Toom-2,3,4 algorithms in $GF(2)$ were described in full details, we can estimate the cost of any one of the steps. Each product of two polynomials would require the cost for evaluation C_e counted twice, the cost for point-wise multiplication C_m, plus the cost for interpolation and recomposition C_i. Since we are dealing with vector-matrix products, where the matrix is fixed *a priori*, we can assume that all evaluations for the fixed operand are pre-computed.

Moreover, when multiplying a k_0 vector by a $k_0 \times n_0$ matrix, we can reduce the count to the strictly needed operations. We assume a pre-computation for all the evaluations of the matrix entries, and we need to evaluate the vector components only once: so we will count only k_0 evaluations. After the $n_0 k_0$ point-wise multiplications, we combine evaluated products $f_i g_i(\alpha), f_j g_j(\alpha), \ldots$ to obtain evaluations for the results like

$$(f_i g_i + f_j g_j + \cdots)(\alpha) \ ,$$

then we interpolate only the n_0 result polynomials: so we count only n_0 interpolations and recompositions.

5.2 Toom-2, also Known as Karatsuba

The Toom-2 algorithm splits the operands in two parts. Starting from two p-bits long polynomials we operate on $\lceil p/2 \rceil$-bits long parts.

Evaluation requires an addition of two parts. The 3 point-wise multiplication operates on parts, and gives doubled parts for results. Interpolation requires $5/2$ additions on such doubled parts. Since an addition of two polynomials in $GF(2)[x]$ with degree d requires d operations, we can conclude that the use of Karatsuba to multiply a degree p polynomial by a fixed one requires:

- $\lceil p/2 \rceil$ operations for the evaluation,
- 3 multiplications of polynomials with degree $\lceil p/2 \rceil$,
- $5\lceil p/2 \rceil$ operations for the interpolation.

5.3 Cost of Exact Divisions

All the Toom-s with splitting order $s > 2$ require some exact divisions. We need to evaluate the cost of this kind of operation.

First of all, we highlight the fact that all the divisions needed for Toom-3 and Toom-4 in $GF(2)[x]$ are divisions by binomials of the form $x^w + 1$, with $w \in \{1, 2, 3\}$. Moreover, all the divisions in the Toom-Cook algorithm are exact divisions [24] and, therefore, can be computed in linear time [25]. When the divisor is in the very special form $x^w + 1$, exact division can be obtained *in-place* with the following simple code.

Input: the degree d, the vector $(a_0, \ldots, a_d)$ of coefficients of the polynomial $A = \sum_{i=0}^{d} a_i x^i$, the divisor in the form $D = x^w + 1$.

Output: the overwritten vector $(a_0, \ldots, a_{d-w})$ of coefficients of the new polynomial $A/D = \sum_{i=0}^{d-w} a_i x^i$.

Execution: for $i = 0 \ldots (d - w)$, $a_{i+w} \leftarrow a_{i+w} + a_i$.

So, a little bit less than d operations are required for any exact division needed in Toom-3 and Toom-4 on $GF(2)$. Since the division process cannot be parallelized as the addition does, we add an extra weight and we double the cost: we consider $2d$ bit operations for each exact division. The given algorithm overwrites its input, so it should be slightly modified for general use; however, all the algorithms presented in [24], and proposed here, works with *in-place* operations. Other operations used in Toom-Cook are bit-shifts, but they can be avoided with word-alignment [23] in software implementation, and they have practically no cost in hardware implementations.

5.4 Toom-3 and Toom-4

From [24], we take the number of basic operations needed for the 3 and 4-way splitting. As usual we consider p-bits operands.

With Toom-3 we operate on $\lceil p/3 \rceil$-bits long parts. Evaluation requires 5 additions and 2 shifts per operand. The 5 point-wise multiplications involve $\lceil (p/3) + 2 \rceil$-bits long parts, because of evaluations in x and $x+1$, which increase the degree. Interpolation operates on doubled parts, and requires 10 additions, 2 shifts and 2 divisions by $x+1$, plus 2 other additions for final recomposition; in total we have a cost of $10 + 2 \cdot 2 + 2$ additions.

The Toom-3 product by a fixed operand hence requires:

- $\lceil p/3 \rceil \cdot 5$ bit operations for the evaluation,
- 5 multiplications of polynomials with degree $\lceil (p/3) + 2 \rceil$,
- $\lceil (p/3) + 2 \rceil \cdot 2 \cdot 17$ bit operations for the interpolation.

For Toom-4 we have $\lceil p/4 \rceil$-bits long parts. Evaluation requires 15 additions and 9 shifts per operand. The 7 point-wise multiplications involve $\lceil (p/4)+3 \rceil$-bits long parts. Interpolation operates on doubled parts, and requires 29 additions, 16 shifts and 4 divisions by x^2+1 and x^3+1, plus 3 other additions for final recomposition; in total we have the cost of $29 + 4 \cdot 2 + 3$ additions.

The Toom-4 product by a fixed operand hence requires:

- $\lceil p/4 \rceil \cdot 15$ bit operations for the evaluation,
- 7 multiplications of polynomials with degree $\lceil (p/4) + 3 \rceil$,
- $\lceil (p/4) + 3 \rceil \cdot 2 \cdot 40$ bit operations for the interpolation.

All Toom-s methods can be used recursively and have asymptotic complexity $O(p^{\log_s(2s-1)})$, but the bigger the splitting order, the heavier the overhead of evaluation/interpolation.

5.5 Numerical Examples

Let us fix the following choice for the system parameters: $p = 8192, k_0 = 2, n_0 = 3$, that are the choices adopted in the second variant of the cryptosystem (see Section 3.3).

The use of 2 recursions of Toom-4, 1 of Toom-3 and 4 of Toom-2 reduces one p-bit multiplication to $7^2 \cdot 5 \cdot 3^4 = 19845$, 11-bits sized, sub-products, each one

with a cost of $11^2/2$ operations. We have an overhead of 301 655 operations for evaluations and 1 617 574 for interpolation. The total cost of computing $\mathbf{u} \cdot \mathbf{G}'$ is then $301\,655 \cdot k_0 + 1\,617\,574 \cdot n_0 + 7^2 \cdot 5 \cdot 3^4 \cdot 11^2 \cdot k_0 \cdot n_0/2 + n_0 \cdot p = 12\,684\,343$ bit operations, included the final reduction modulo (x^p+1). This count gives a far smaller result than the naïve technique, that requires $k_0 \cdot n_0 \cdot p^2/2 = 201\,326\,592$ operations. With the same approach, we can obtain the cost of computing $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$: 8 657 332 operations using Toom-Cook, versus 134 217 728 with a naïve implementation.

With parameters $p = 4096, k_0 = 3, n_0 = 4$, that are very similar to the choices adopted for the proposal in [9], we assume 3 recursions of Toom-4 and 3 of Toom-2, that result in a number of bit operations for $\mathbf{u} \cdot \mathbf{G}'$ of 8 074 444 versus 100 663 296 with the naïve approach. While for the decryption step $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$ we obtain 6 166 127 bit operations versus 75 497 472.

6 Vector-Toeplitz Convolution

Another approach to speed-up polynomial products in cryptography is the Winograd convolution [26]. It is very similar to Karatsuba's multiplication, but it has a smaller overhead. We shortly recall it. Given an even sized $2d \times 2d$ Toeplitz matrix $\mathbf{T}$, we can *factorize* it as follows:

$$\begin{pmatrix} \mathbf{T}_0 & \mathbf{T}_1 \\ \mathbf{T}_2 & \mathbf{T}_0 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{T}_1 - \mathbf{T}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_2 - \mathbf{T}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T}_0 \end{pmatrix} \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} \end{pmatrix} ,$$

where $\mathbf{I}$ is the $d \times d$ identity matrix, and $\mathbf{T}_0, \mathbf{T}_1, \mathbf{T}_2$ are themselves $d \times d$ Toeplitz matrices, as also $\mathbf{T}_1 - \mathbf{T}_0$ and $\mathbf{T}_2 - \mathbf{T}_0$. It follows that the vector-matrix product $(\mathbf{V}_0, \mathbf{V}_1) \cdot \mathbf{T}$ can be computed with three steps:

- the addition $\mathbf{V}_0 + \mathbf{V}_1$,
- 3 vector-matrix sub-products by $d \times d$ Toeplitz matrices,
- 2 more additions to obtain the result.

Since circulant matrices are also Toeplitz and the proposed size p is a power of 2, this optimization can be used for as many recursions as needed for our computations. Asymptotic complexity is exactly the same for Toom-2 and this approach, but if we analyze again the pre- and post-computation, we obtain:

- $p/2$ operations for the "evaluation",
- 3 multiplications with dimension $p/2$,
- $p/2$ operations for the "interpolation".

6.1 Numerical Examples

This method cannot be mixed with Toom-Cook, and its main advantage is that it is much easier to implement in software.

If we consider again $p = 8192, k_0 = 2, n_0 = 3$, the use of 11 recursions of Winograd's method leads to 14 106 224 operations for encryption versus 12 684 343

obtained with Toom. The decryption step $\mathbf{u}' \cdot \mathbf{S} = \mathbf{u}$ can be completed in 9 871 080 operations, around 15% more than with the polynomial strategy, but with a far simpler source code. With smaller parameters, for example $p = 4096, k_0 = 3, n_0 = 4$, the difference is smaller: 11 recursions used for encryption give 8 103 706 operations, only 1% more than the cost computed using Toom-Cook.

On the other hand, the use of polynomials gives greater flexibility. For example, with the odd parameter $p = 5555, k_0 = 2, n_0 = 3$, 2 recursions of Toom-4 and 5 of Toom-2 give a cost for encryption of 7 310 809 bit operations, a 12× speed-up with respect to the naïve approach. For the same parameters, Winograd's trick is not applicable at all, because p is odd.

7 Cryptosystem Complexity Assessment

In this section we evaluate the encryption and decryption complexity of the proposed cryptosystem, by considering the usage of efficient computation algorithms suitable for the particular structure of the matrices involved.

Encryption complexity is due to multiplication of the cleartext by the code generator matrix and to addition of intentional errors. It can be expressed as follows:

$$C_{enc} = C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right) + n \tag{12}$$

where $C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right)$ represents the number of operations needed for calculating the product $\mathbf{u} \cdot \mathbf{G}'$ and n binary operations are considered for the addition of vector $\mathbf{e}$.

The decryption complexity, instead, can be divided into three parts:

$$C_{dec} = C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right) + C_{SPA} + C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right) \tag{13}$$

where $C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right)$ and $C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right)$ represent the number of operations needed for computing $\mathbf{x} \cdot \mathbf{Q}$ and $\mathbf{u}' \cdot \mathbf{S}$, respectively, while C_{SPA} is the number of operations required for LDPC decoding through the sum-product algorithm. In expressions (12) and (13), $C_{mul}\left(\mathbf{u} \cdot \mathbf{G}'\right)$ and $C_{mul}\left(\mathbf{u}' \cdot \mathbf{S}\right)$ involve multiplication by dense matrices, so we can resort to efficient algorithms, like the Toom-Cook method described in the previous section. $C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right)$, instead, expresses the number of operations needed to perform the product of a $1 \times n$ vector by a sparse $n \times n$ matrix ($\mathbf{Q}$, with row/column weight equal to m). For this reason, we resort to the naïve implementation, that has the lowest complexity $C_{mul}\left(\mathbf{x} \cdot \mathbf{Q}\right) = n \cdot m$.

For the decoding complexity, the following expression can be adopted [16]:

$$C_{SPA} = I_{ave} \cdot n \left[q\left(8d_v + 12R - 11\right) + d_v\right] \tag{14}$$

where I_{ave} is the average number of decoding iterations, q is the number of quantization bits used inside the decoder and $R = k_0/n_0$ is the code rate. Numerical simulations have permitted to verify that, for the codes involved in the first cryptosystem implementation, assuming $q = 6$ and $t = 190$, it is $I_{ave} \simeq 5$.

For the codes used in the new variant, instead, assuming $q = 6$ and $t = 470$, it is $I_{ave} \simeq 9$. These values of I_{ave} are further reduced for smaller t.

As concerns the public key length, the proposed cryptosystem uses, as the public key, a generator matrix, $\mathbf{G}'$, formed by $k_0 \times n_0$ circulant blocks with size p. Therefore, it can be completely described by $k_0 \cdot n_0 \cdot p$ bits.

Table 1. Comparison between the proposed versions of QC-LDPC based McEliece cryptosystem and other schemes

	McEliece (original)	Niederreiter	RSA	QC-LDPC McEliece 1	QC-LDPC McEliece 2
Key Size (bytes)	67072	32750	256	6144	6144
Information Bits	524	276	1024	12288	16384
Transmission Rate	0.5117	0.5681	1	0.75	0.6667
Enc Ops per bit	514	50	2402	658	776
Dec Ops per bit	5140	7863	738 112	4678	8901

Table 1 reports the characteristics of the two proposed variants of McEliece cryptosystem based on QC-LDPC codes, both secure against the known attacks. The first variant (noted as QC-LDPC McEliece 1) adopts the choice of the system parameters we have already proposed in [9], with the only difference of $p = 4096$ instead of 4032. We have considered p coincident with a power of two in this version because, for such values, a circulant matrix with odd row/column weight is always non-singular, as shown in the Appendix A. The choice of $p = 4096$ instead of 4032, however, has no effect on the system security. In the second variant (noted as QC-LDPC McEliece 2), the system parameters have been changed in order to increase the system security level. For the sake of comparison, also the original McEliece, the Niederreiter and the RSA cryptosystems are considered. The key length for the Niederreiter cryptosystem coincides with the number of bits in the non-systematic part of matrix $\mathbf{H}$.

From the security viewpoint, these systems are not equivalent: the first proposal exhibits a security level of 2^{71} binary operations, while the second one exceeds the threshold of 2^{80} binary operations, that is currently considered as an up to date technology limit. The first three solutions are instead assumed with their standard parameters [4]. In such case, the McEliece and Niederreiter cryptosystems are not able to reach a similar security level; however, more secure versions would yield increased complexity.

It results from the table that both the proposed variants of McEliece cryptosystem based on QC-LDPC codes represent a trade-off between the original McEliece cryptosystem (and its Niederreiter version) and other cryptosystems, like RSA. In fact, they represent an advance in overcoming the drawbacks of the original McEliece cryptosystem: they have very smaller public keys and increased transmission rate. With respect to RSA, the proposed cryptosystems have the advantage of very lower complexity, that is only slightly increased with respect to the original McEliece version (that, moreover, has a lower security level).

8 Conclusions

We have elaborated on an implementation of the McEliece cryptosystem based on QC-LDPC codes we have proposed for overcoming the main drawbacks of its original version, that has been recently discovered to be subject to dangerous attacks. We have described how these attacks exploit the sparse character of some constituent matrices, together with their diagonal form, and we have proposed two new variants of the cryptosystem that do not allow the application of such attack techniques. As typical in cryptography, this does not exclude that further attacks might be conceived in the future. So, an effort should be made for getting a coding based cryptographic construction with a supporting proof of security, similar to what done in the related area of lattice based cryptography [27]. For the time being, based on our knowledge, we can say that possible progress in cryptanalysis of the proposed system will require the definition of substantially new strategies.

We have also reported complexity estimates based on the Toom-Cook method for polynomials in $GF(2)[x]$. They can be partially extended to other cryptosystems, such as NTRU, where polynomials modulo $(x^n \pm 1)$ are used, and ECC on $GF(2^n)$. For both these systems, the use of Karatsuba's and Winograd's fast convolution were proposed [28,29], and the Toom-Cook method could be applied as well, even if its effect should be not as impressive as in the proposed system.

The promising results obtained with Toom-Cook have their main reason in the use of matrices. The additional cost of fast multiplication methods is quite big, that is the reason why they are effective only for big operands. Numerical examples given in Section 5.5 show that, for deep recursion, more than half the cost of a single product comes from evaluation and interpolation. This cost, however, is reduced in the proposed cryptosystem, because only a few evaluations and interpolations are needed for a set of multiplications, so that deepest recursion and biggest saving are possible.

The application of the Toom-Cook method permits to reduce the encryption and decryption complexity of the proposed cryptosystems that, furthermore, are able to overcome the main drawbacks of the original McEliece cryptosystem in terms of key size and transmission rate. For these reasons, they can be seen as a valuable trade-off between the original McEliece cryptosystem and other widespread solutions, like RSA.

References

1. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. DSN Progress Report, 114–116 (1978)
2. Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Trans. Inform. Theory 24, 384–386 (1978)
3. Lee, P., Brickell, E.: An observation on the security of McEliece's public-key cryptosystem. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 275–280. Springer, Heidelberg (1988)

4. Canteaut, A., Chabaud, F.: A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. IEEE Trans. Inform. Theory 44, 367–378 (1998)
5. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Contr. and Inform. Theory 15, 159–166 (1986)
6. Li, Y.X., Deng, R., Wang, X.M.: On the equivalence of McEliece's and Niederreiter's public-key cryptosystems. IEEE Trans. Inform. Theory 40, 271–273 (1994)
7. Riek, J.: Observations on the application of error correcting codes to public key encryption. In: Proc. IEEE International Carnahan Conference on Security Technology. Crime Countermeasures, Lexington, KY, USA, October 1990, pp. 15–18 (1990)
8. Richardson, T., Urbanke, R.: The capacity of low-density parity-check codes under message-passing decoding. IEEE Trans. Inform. Theory 47, 599–618 (2001)
9. Baldi, M., Chiaraluce, F.: Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC codes. In: Proc. IEEE ISIT 2007, Nice, France, June 2007, pp. 2591–2595 (2007)
10. Monico, C., Rosenthal, J., Shokrollahi, A.: Using low density parity check codes in the McEliece cryptosystem. In: Proc. IEEE ISIT 2000, Sorrento, Italy, June 2000, p. 215 (2000)
11. Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes. In: Proc. First International Conference on Symbolic Computation and Cryptography (SCC 2008), Beijing, China (April 2008)
12. Gaborit, P.: Shorter keys for code based cryptography. In: Proc. Int. Workshop on Coding and Cryptography (WCC 2005), Bergen, Norway, March 2005, pp. 81–90 (2005)
13. Richardson, T., Urbanke, R.: Efficient encoding of low-density parity-check codes. IEEE Trans. Inform. Theory 47, 638–656 (2001)
14. Neal, R.M.: Faster encoding for low-density parity check codes using sparse matrix methods (1999), `http://www.cs.toronto.edu/~radford/ftp/ima-part1.pdf`.
15. Stern, J.: A method for finding codewords of small weight. In: Wolfmann, J., Cohen, G. (eds.) Coding Theory 1988. LNCS, vol. 388, pp. 106–113. Springer, Heidelberg (1989)
16. Baldi, M., Chiaraluce, F.: LDPC Codes in the McEliece Cryptosystem (September 2007), `http://arxiv.org/abs/0710.0142`
17. Karatsuba, A.A., Ofman, Y.: Multiplication of multidigit numbers on automata. Soviet Physics Doklady 7, 595–596 (1963)
18. Toom, A.L.: The complexity of a scheme of functional elements realizing the multiplication of integers. Soviet Mathematics Doklady 3, 714–716 (1963)
19. Cook, S.A.: On the minimum computation time of functions. PhD thesis, Dept. of Mathematics, Harvard University (1966)
20. Bodrato, M., Zanoni, A.: Integer and polynomial multiplication: Towards optimal Toom-Cook matrices. In: Brown, C.W. (ed.) Proceedings of the ISSAC 2007 Conference, July 2007, pp. 17–24. ACM Press, New York (2007)
21. Cantor, D.G.: On arithmetical algorithms over finite fields. Journal of Combinatorial Theory A 50, 285–300 (1989)
22. Schönhage, A.: Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. Acta Informatica 7, 395–398 (1977)
23. Brent, R.P., Zimmermann, P., Gaudry, P., Thomé, E.: Faster multiplication in GF(2)[x]. In: van der Poorten, A.J., Stein, A. (eds.) ANTS-VIII 2008. LNCS, vol. 5011, pp. 153–166. Springer, Heidelberg (2008)

24. Bodrato, M.: Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 116–133. Springer, Heidelberg (2007)
25. Jebelean, T.: An algorithm for exact division. Journal of Symbolic Computation 15, 169–180 (1993)
26. Winograd, S.: Arithmetic Complexity of Computations. CBMS-NSF Regional Conference Series in Mathematics, vol. 33. SIAM, Philadelphia (1980)
27. Micciancio, D.: Generalized compact knapsacks, cyclic lattices and efficient one-way functions. Computational Complexity 16, 365–411 (2007)
28. Silverman, J.H.: High-speed multiplication of (truncated) polynomials. Technical Report 10, NTRU CryptoLab (January 1999)
29. Weimerskirch, A., Stebila, D., Shantz, S.C.: Generic GF(2) arithmetic in software and its application to ECC. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 79–92. Springer, Heidelberg (2003)

A Matrix Inversion in $GF(2)[x]/(x^p + 1)$

For whatever choice of p, we work on the polynomial ring $\mathbb{R} = GF(2)[x]/(x^p+1)$. In any case $x^p + 1$ is divisible by $x + 1$, because we are working in characteristic 2, so that the ring is not a field: it has zero divisors and non invertible elements.

If the chosen p is a power of two ($p = 2^a$), then $x + 1$ is the only prime factor of $x^p + 1 \equiv (x + 1)^p$ and, in such a case, it is very easy to check if an element in $\mathbb{R}$ is not invertible. In fact, it suffices checking if 1 is a root, or, equivalently, if the number of non-zero (1) coefficients is even.

If p is not a power of two, the invertibility check becomes more involved. Obviously general theorems are still valid, so that we can say that a generic element $f \in \mathbb{R}$ is invertible if and only if it is coprime with $x^p + 1$, and a matrix is invertible if and only if its determinant is invertible. But invertible matrices can exist that only contain non invertible entries.

So one needs a clever algorithm to compute the inverses of the matrices, either by computing the inverse on any sub-ring $GF(2)[x]/d^\alpha$ where $d^\alpha | x^p + 1$, then combining the results with the Chinese Remainder Theorem, or by a modified version of Gaussian inversion, exploiting Bezout's identity to obtain a pivot on columns without invertible elements.

Full Cryptanalysis of LPS and Morgenstern Hash Functions

Christophe Petit[1,⋆], Kristin Lauter[2], and Jean-Jacques Quisquater[1,⋆⋆]

[1] UCL Crypto Group[⋆⋆⋆]
[2] Microsoft Research
christophe.petit@uclouvain.be, klauter@microsoft.com, jjq@uclouvain.be

Abstract. Collisions in the LPS cryptographic hash function of Charles, Goren and Lauter have been found by Zémor and Tillich [17], but it was not clear whether computing preimages was also easy for this hash function. We present a probabilistic polynomial time algorithm solving this problem. Subsequently, we study the Morgenstern hash, an interesting variant of LPS hash, and break this function as well. Our attacks build upon the ideas of Zémor and Tillich but are not straightforward extensions of it. Finally, we discuss fixes for the Morgenstern hash function and other applications of our results.

1 Introduction

Hash functions are widely used in cryptographic applications such as commitment schemes, digital signatures schemes, message authentication codes or password encryption. Typically, a hash function is required to be preimage and collision resistant and to have nearly uniform output distribution. Due to the importance of cryptographic hash functions, the SHA family was designed as a NIST standard [2]. However, recently discovered vulnerabilities in SHA-1 [15] prompted NIST to launch a competition for a New Cryptographic Hash Algorithm [1].

The NIST competition is stimulating research on hash functions in the cryptographic community and a lot of new schemes have been recently designed and put forward. Particularly appealing from a theoretical point of view, some of these schemes are *provably secure*, in the sense that their security relates to the hardness of some mathematical problem [8,4,3,13]. A good reduction to a simply formulated mathematical challenge facilitates the evaluation process and increases the confidence once the function has resisted first cryptanalytic attempts. However, it also gives the cryptanalyst a clue to break the scheme, and is especially problematic if the mathematical challenge turns out to be easy.

⋆ Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS). Part of this work was done while visiting the Security and Cryptography group of the Computer Science Department, UCSD.

⋆⋆ Part of this work was done while visiting MIT (CSAIL-Theory of Computation).

⋆⋆⋆ A member of BCRYPT and ECRYPT networks.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 263–277, 2008.

The LPS hash function proposed by Charles, Goren and Lauter is one of these constructions [3]. It has a particularly elegant design, as the hash computation can be interpreted as a random walk in the optimal expander graphs of Lubotzky, Philips and Sarnak [7]. Finding collisions for this function is finding cycles in the graphs, which also amounts to finding a non-trivial factorization of the identity in terms of some particular elements of a projective group of matrices (a problem we will call the *decomposition problem*). Charles, Goren and Lauter proposed this problem as potentially hard. A major step in the breaking of the LPS hash function has recently been performed by Zémor and Tillich [17] who produced collisions by actually solving this problem.

One of the main contributions of this paper is an efficient algorithm that finds preimages for the LPS hash function. As Zémor and Tillich did, we actually solve the underlying problem which was presumed hard. Both for efficiency considerations and because of these new attacks, it also seemed worth studying the Morgenstern hash function, an interesting variant of the LPS hash relying on different graphs. A second main contribution of this paper is to adapt the Zémor and Tillich attack to Morgenstern hashes, and as an example we give an efficient collision finding algorithm. Combining the ideas of our two algorithms also gives a preimage finding algorithm for Morgenstern hashes that we do not present here due to space limitations.

The paper is organized as follows: in Section 2 we describe the LPS and Morgenstern hash functions; in Section 3 we recall the Zémor and Tillich algorithm; Section 4 presents our preimage algorithm for LPS hashes; in Section 5 we adapt Zémor and Tillich's algorithm to Morgenstern hashes and in Section 6 we discuss fixes for LPS and Morgenstern hashes, as well as potential applications of our results. In the appendix we give toy examples of our algorithms (1024−bit examples are given in the full version of this paper [10]).

2 LPS and Morgenstern Hash Functions

A *Cayley graph* $\mathcal{C}_{G,S} = (V, E)$ is a graph constructed from a group G and a subset S of G as follows: V contains a vertex v_g associated to each element $g \in G$, and E contains the directed edge (v_{g_1}, v_{g_2}) iff there is some $s \in S$ such that $g_2 = g_1 s$. The elements of S are called the *graph generators*. The graph $\mathcal{C}_{G,S}$ is $|S|$-regular; it is connected iff S generates G; it is undirected iff $S = S^{-1}$.

A general construction for a cryptographic hash function from a Cayley graph was introduced by Zémor and Tillich [16,13,14] in the directed case and by Charles, Goren and Lauter [3] in the undirected case. In this paper, we focus on two instances of the undirected graph construction, which we now recall following mainly the description given in [17].

Let $a := |S| - 1$. We will define a function π which orders the set of generators (minus one generator to avoid back-tracking). Fix a function $\pi : \{0, 1...a-1\} \times S \to S$ such that for any $g \in S$ the set $\pi(\{0, 1...a-1\} \times \{g\})$ is equal to $S \setminus \{g^{-1}\}$. Let g_0 and g_{IV} be arbitrary fixed elements of S and G respectively. The input message is converted to a base a number $x_1...x_k$ and the elements $g_i = \pi(x_i, g_{i-1})$

are computed recursively. The hashcode of the input message is the product of group elements $H(x) = g_{IV} g_1 ... g_k$.

We will call hash functions constructed following this design strategy *Cayley hashes*. These hash functions have some very interesting properties:

- The girth of the Cayley graph is the length of the smallest cycle, and no two distinct messages of the same length can collide if their length is less than half the girth.
- If the chosen graphs are good expanders (see [6] for precise definitions and applications), the outputs tend to be uniformly distributed, and the convergence to the uniform distribution is fast.
- Differential cryptanalysis (DC), which has been the most successful approach against SHA-1, does not seem to apply to Cayley hashes. Indeed, DC typically activates various portions of the message simultaneously, while in Cayley hashes the bits (or k-its) are processed one at the time.
- Collision resistance is equivalent to the hardness of a simply-stated representation problem in the corresponding group: namely, this problem is to find a factorization of the identity $1 = g_1 g_2 ... g_t$ with $g_i \in S$ and $g_i g_{i+1} \neq 1$ for all $i \in \{1, 2, ...t-1\}$.
- Preimage resistance and second preimage resistance follow from similar problems.

One proposal by Charles, Goren and Lauter [3] is to use the celebrated LPS graphs of Lubotzky, Philips and Sarnak [7] that we now describe. Let p and l be primes, l small and p large, both p and l equal to 1 mod 4, and l being a quadratic residue modulo p. To l and p is associated an LPS graph $X_{l,p}$ as follows. Let $\mathbf{i}$ be an integer such that $\mathbf{i}^2 \equiv -1 \bmod p$. The vertices of $X_{l,p}$ are elements in the group $G = PSL(2, \mathbb{F}_p)$ (*i.e.* 2×2 matrices of square determinant, modulo the equivalence relation $M_1 \sim \lambda M_2, \lambda \in \mathbb{F}_p^*$). The set S is $S = \{g_j\}_{j=1...,l+1}$, where

$$g_j = \begin{pmatrix} \alpha_j + \mathbf{i}\beta_j & \gamma_j + \mathbf{i}\delta_j \\ -\gamma_j + \mathbf{i}\delta_j & \alpha_j - \mathbf{i}\beta_j \end{pmatrix}, \qquad j = 1, ..., l+1;$$

and $(\alpha_j, \beta_j, \gamma_j, \delta_j)$ are all the integer solutions of $\alpha^2 + \beta^2 + \gamma^2 + \delta^2 = l$, with $\alpha > 0$ and β, γ, δ even. The Cayley graph $X_{l,p} = \mathcal{C}_{G,S}$ is undirected since S is stable under inversion.

The choice of LPS graphs was very appealing : they are Ramanujan (*i.e.*, they have optimal expansion properties asymptotically [7,6]); they have no short cycles, and computing the resulting hash functions turned out to be quite efficient compared to other provable hashes [11]. Unfortunately, it turns out that LPS hash function is neither collision nor preimage resistant (see Sections 3 and 4 below).

For efficiency reasons, we recently considered the use of Morgenstern graphs to replace LPS graphs in Charles-Goren-Lauter's construction [11]. Morgenstern's Ramanujan graphs [9] generalize LPS graphs from an odd prime $p \equiv 1 \bmod 4$ to any power of any prime q. More specifically, we suggested the use of Morgenstern graphs with $q = 2^k$, that we now describe.

Let q be a power of 2 and $f(x) = x^2+x+\epsilon$ irreducible in $\mathbb{F}_q[x]$. Let $p(x) \in \mathbb{F}_q[x]$ be irreducible of even degree $n = 2d$ and let $\mathbb{F}_{q^n}$ be represented by $\mathbb{F}_q[x]/(p(x))$. The vertices of the Morgenstern graph Γ_q are elements of $G = PSL_2(\mathbb{F}_{q^n})$ (*i.e.* 2×2 matrices modulo the equivalence relation $M_1 \sim \lambda M_2, \lambda \in \mathbb{F}_{q^n}^*$). Let $\mathbf{i} \in \mathbb{F}_{q^n}$ be a root of $f(x)$. The set S is taken to be $S = \{g_j\}_{j=1...,q+1}$, where

$$g_j = \begin{pmatrix} 1 & \gamma_j + \delta_j \mathbf{i} \\ (\gamma_j + \delta_j \mathbf{i} + \delta_j)x & 1 \end{pmatrix}, \qquad j = 1, ..., q+1;$$

where $\gamma_j, \delta_j \in \mathbb{F}_q$ are all the $q+1$ solutions in $\mathbb{F}_q$ for $\gamma_j^2 + \gamma_j\delta_j + \delta_j^2\epsilon = 1$. The Cayley graphs $\Gamma_q = \mathcal{C}_{G,S}$ are also undirected as each g_j has order 2.

An interesting property of Morgenstern hashes compared to LPS hashes is that arithmetic is done in fields that are extensions of $\mathbb{F}_2$ rather than in finite prime fields, potentially leading to faster hashes for some architectures. The total break of LPS hashes leads to the question of whether similar attacks can be found for Morgenstern hashes. This is indeed the case, and as an example we give a collision-finding attack for $q = 2$ in Section 5.

3 Zémor and Tillich Algorithm

As our new attacks will build upon it, we now briefly recall Zémor and Tillich's algorithm that finds collisions for LPS hashes [17]. The algorithm lifts the graph generators and the representation problem from $PSL(2, \mathbb{F}_p)$ to an appropriate subset Ω of $SL(2, \mathbb{Z}[i])$ (in this section and the next one, i is the complex imaginary number satisfying $i^2 + 1 = 0$ while $\mathbf{i}$ is a solution to $\mathbf{i}^2 + 1 \equiv 0 \bmod p$). The relevant set is

$$\Omega = \left\{ \begin{pmatrix} a+bi & c+di \\ -c+di & a-bi \end{pmatrix} | (a, b, c, d) \in E_e \text{ for some integer } e > 0 \right\}$$

where E_e is the set of 4-tuples $(a, b, c, d) \in \mathbb{Z}^4$ such that

$$\begin{cases} a^2 + b^2 + c^2 + d^2 = l^e \\ a > 0, a \equiv 1 \bmod 2 \\ b \equiv c \equiv d \equiv 0 \bmod 2. \end{cases}$$

We will call the first of these equations describing E_e the *norm equation*, as the left-hand side of this equation is the norm of the quaternion corresponding to the quadruplet (a, b, c, d) (see [7]). The set Ω has two important properties: first, any element of Ω admits a unique factorization in terms of the lifts of the graph generators, and second, there exists a multiplicative homomorphism from Ω to $PSL(2, \mathbb{F}_p)$ that allows translation of this factorization back to $PSL(2, \mathbb{F}_p)$.

In their exposition, Zémor and Tillich decompose their attack into three steps. The first step (lifting the decomposition problem to $SL(2, \mathbb{Z}[i])$) amounts to finding integers a, b, c, d and λ satisfying the following conditions:

$$\begin{cases} (a, b, c, d) \in E_e \\ (a, b, c, d) \quad \text{not divisible by l} \\ (a, b, c, d) \equiv \lambda(1, 0, 0, 0) \bmod p. \end{cases}$$

Putting every congruence condition into the norm equation leads to a diophantine equation that was solved by Zémor and Tillich in their paper. The second step of the attack is to factorize the lifted element I' of Ω into products of lifted generators $g'_j, j = 1...l+1$. We know this factorization is unique and has size e, so let us write it $I' = g'_{j_1} g'_{j_2} ... g'_{j_e}$. Multiplying on the right by a lifted generator g' gives a matrix that is divisible by l if and only if $g' = (g'_{j_e})^{-1}$, so by trying each of the graph generators we get the last factor, and we then proceed recursively. The final step is to transpose the factorization of I' in Ω into a factorization of the identity in $PSL(2, \mathbb{F}_p)$, but using the homomorphism from Ω to $PSL(2, \mathbb{F}_p)$, this last step is trivial. For details on the attack we refer to [17].

4 Finding Preimages for LPS Hashes

Suppose we are given a matrix $M = \begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix} \in PSL(2, \mathbb{F}_p)$ which has square determinant, and we are asked to find a preimage, that is a factorization of it with the graph generators. By solving two linear equations in $\mathbb{F}_p$ we can write it in the form

$$M = \begin{pmatrix} A + B\mathbf{i} & C + D\mathbf{i} \\ -C + D\mathbf{i} & A - B\mathbf{i} \end{pmatrix}.$$

Our algorithm follows along the lines of Zémor and Tillich's. We first lift the problem from $PSL(2, \mathbb{F}_p)$ to the set Ω defined above, then factorize in Ω and finally come back to $PSL(2, \mathbb{F}_p)$. The only difference will be in the first step. Lifting the representation problem now amounts to finding integers a, b, c, d and λ satisfying the following conditions:

$$\begin{cases} (a, b, c, d) \in E_e \\ (a, b, c, d) \quad \text{not divisible by l} \\ (a, b, c, d) \equiv \lambda(A, B, C, D) \bmod p. \end{cases}$$

We write $a = A\lambda + wp$, $b = B\lambda + xp$, $c = C\lambda + yp$ and $d = D\lambda + zp$ with $w, x, y, z \in \mathbb{Z}$. For convenience we choose e even, that is $e = 2k$ for k an integer. The norm equation becomes

$$(A\lambda + wp)^2 + (B\lambda + xp)^2 + (C\lambda + yp)^2 + (D\lambda + zp)^2 = l^{2k}. \tag{1}$$

In the case $B = C = D = 0$ the norm equation is $(A\lambda + wp)^2 + (xp)^2 + (yp)^2 + (zp)^2 = l^{2k}$ and was solved by Zémor and Tillich as follows: choose

$$A\lambda + wp = l^k + mp^2$$

for small m and appropriate k, hence the equation is already satisfied modulo p^2. Simplifying by p^2 we get a quadratic diophantine equation of type $x^2 + y^2 + z^2 = m(l^k - mp^2)$ which Zémor and Tillich show has a solution either for $m = 1$ or for $m = 2$. In Equation 1, when B, C, D are non-zero we cannot divide by p^2 because of the term $2p(wA + xB + yC + zD)\lambda$. Since we do not, the coefficients

of degree-2 terms are huge (at least p), and the equation is at first sight very hard to solve.

We overcome this difficulty with a new idea. In the remainder of this section, we will solve the preimage problem for diagonal matrices with A and/or B non-zero, and then we will write any matrix as a product of four diagonal matrices and up to four graph generators. Altogether this leads to an efficient probabilistic algorithm that finds preimages of the LPS hash function.

Preimages for diagonal matrices. Now we show how to find a factorization of a matrix

$$M = \begin{pmatrix} A + B\mathbf{i} & \\ & A - B\mathbf{i} \end{pmatrix}$$

such that $A^2 + B^2$ is a square modulo p. Write $y = 2y'$ and $z = 2z'$ where y', z' are integers. We need to find integer solutions to

$$\begin{cases} (A\lambda + wp)^2 + (B\lambda + xp)^2 + 4p^2(y'^2 + z'^2) = l^{2k} \\ A\lambda + wp \equiv 1 \bmod 2 \\ B\lambda + xp \equiv 0 \bmod 2 \end{cases}$$

Fix $k = \lceil \log_l(8p^2) \rceil$. As $A^2 + B^2$ is a square, there are exactly two values for λ in $\{0, 1, ...p-1\}$ satisfying the norm equation modulo p:

$$(A^2 + B^2)\lambda^2 = l^{2k} \bmod p.$$

Choose either of them, and let $m := (l^{2k} - (A^2 + B^2)\lambda^2)/p$. Our strategy will be to pick random solutions to

$$\begin{cases} l^{2k} - (A\lambda + wp)^2 - (B\lambda + xp)^2 \equiv 0 \bmod p^2 \\ A\lambda + wp \equiv 1 \bmod 2 \\ B\lambda + xp \equiv 0 \bmod 2 \end{cases}$$

until the equation

$$4(y'^2 + z'^2) = n$$

has solutions, where

$$n := \left(l^{2k} - (A\lambda + wp)^2 - (B\lambda + xp)^2\right)/p^2.$$

A random solution to the congruence system is computed as follows: until you get x with the correct parity, pick a random $w \in \{0, 1, ...p-1\}$ with the right parity and compute $x = \frac{m}{2\lambda B} - \frac{A}{B}w \bmod p$. By the way k, x and w are chosen we are guaranteed that $n > 0$ so the equation $4(y'^2 + z'^2) = n$ has solution if and only if 4 divides n and all prime factors of n congruent to 3 modulo 4 appear an even number of times in the factorization of n. To avoid the factorization of n in the algorithm, we will actually strengthen this condition to n being equal to 4 times a prime congruent to 1 modulo 4. When it has solutions, the equation $4(y'^2 + z'^2) = n$ is easily solved with the Euclidean algorithm, as recalled in [17]. After lifting the problem to $SL(2, \mathbb{Z}[i])$ the second and third steps of the algorithm are the same as in Zémor-Tillich algorithm. So we are done with the factorization of diagonal matrices.

Reduction to the diagonal case. Now we show how to decompose any matrix $M \in PSL(2, \mathbb{F}_p)$ into a product of diagonal matrices and graph generators. We may additionally assume that all the entries of M are nonzero: if they are not, just multiply M by gg^{-1} for some adequate g in S, and consider the factorization of $g^{-1}M$. We will show how to find $(\lambda,\ \alpha,\ \omega,\ \beta_1,\ \beta_2)$ with the last four being squares, such that

$$\begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix} = \lambda \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix} \begin{pmatrix} f_1 & f_2 \\ f_3 & f_4 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} = \lambda \begin{pmatrix} f_1 & \omega f_2 \\ \alpha f_3 & \alpha\omega f_4 \end{pmatrix} \tag{2}$$

and

$$\begin{aligned} \begin{pmatrix} f_1 & f_2 \\ f_3 & f_4 \end{pmatrix} &= \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \beta_1 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \beta_2 \end{pmatrix} \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 - 4\beta_1 - 4\beta_2 - 4\beta_1\beta_2 & 2 - 8\beta_1 + 2\beta_2 + 2\beta_1\beta_2 \\ -2 - 2\beta_1 + 8\beta_2 - 2\beta_1\beta_2 & -4 - 4\beta_1 - 4\beta_2 + \beta_1\beta_2 \end{pmatrix}. \end{aligned}$$

Lemma 1. *Matrix equation (2) is equivalent to the following system:*

$$\begin{cases} M_2M_3f_1f_4 - M_1M_4f_2f_3 = 0 \\ \alpha M_1f_3 - M_3f_1 = 0 \\ \omega M_3f_4 - M_4f_3 = 0 \\ \lambda f_1 - M_1 = 0 \end{cases} \tag{3}$$

Proof : ($\Rightarrow$) Fourth equation is entry (1,1) of the matrix equation. Third equation is entry (2,1) times M_1 minus entry (1,1) times M_3. Second equation is entry (1,2) times M_1 minus entry (1,1) times M_2. First equation is entry (1,1) times entry (2,2) times M_2M_3 minus entry (1,2) times entry (2,1) times M_1M_4.
($\Leftarrow$) Last equation is $M_1 = \lambda f_1$ that is entry (1,1). We have $M_2 = \frac{M_1M_4f_2f_3}{M_3f_1f_4}$ by first equation so $M_2 = f_2\frac{M_4f_3}{M_3f_4}\frac{M_1}{f_1} = f_2\omega\lambda$ by third and fourth equation, that is entry (1,2). We have $M_3 = \frac{\alpha M_1f_3}{f_1} = \alpha\lambda f_3$ by second then fourth equation, that is entry (2,1). We have $M_4 = \omega M_3\frac{f_4}{f_3}$ by third equation, so using the already proved entry (2,1) we have $M_4 = \omega\alpha\lambda f_3\frac{f_4}{f_3} = \omega f_4\alpha\lambda$ that is entry (2,2). □

In the system of equations (3), the first equation only involves β_1 and β_2 while the other equations are linear once β_1 and β_2 are fixed. So we can concentrate on solving the first equation, which is quadratic in both β_1 and β_2:

$$\begin{aligned} M_2M_3f_1f_4 - M_1M_4f_2f_3 = {} & 4(M_2M_3 - M_1M_4)(-\beta_1^2 + 3\beta_1 + 4)\beta_2^2 \\ & + \left(M_2M_3(12\beta_1^2 + 49\beta_1 + 12) + M_1M_4(-12\beta_1^2 + 76\beta 1 - 12)\right)\beta_2 \\ & + 4(M_2M_3 - M_1M_4)\left(4\beta_1^2 + 3\beta_1 - 1\right). \end{aligned}$$

Our algorithm then proceeds as follows:

1. Pick a random β_1 which is a square.
2. Compute the discriminant of the quadratic equation in β_2, β_1. If it is not a square, go back to 1.

3. Solve the quadratic equation. If none of the roots is a square, go back to 1. Else, assign a quadratic root to β_2.
4. Compute f_1, f_2, f_3, f_4.
5. Solve $\alpha M_1 f_3 - M_3 f_1 = 0$ to get α. If α is not a square, go back to 1.
6. Solve $\omega M_3 f_4 - M_4 f_3 = 0$ to get ω. If ω is not a square, go back to 1.

This concludes the exposition of our algorithm.

Runtime analysis. First consider the algorithm for diagonal matrices. Assuming n behaves "as a random number" then according to the prime number theorem we will need $\mathcal{O}(\log n) = \mathcal{O}(\log p)$ trials before getting one n of the correct form. For each trial, the most expensive computation is a primality test, which can be done in polynomial time (in our implementation, we actually use the probabilistic function *mpz_probab_prime_p* of GNU MP). So the algorithm for diagonal matrices is probabilistic polynomial time. In the reduction algorithm, the probability for a random number to be a square modulo p being one half, we estimate that a solution $(\lambda, \alpha, \omega, \beta_1, \beta_2)$ with the last four being squares can be found in about 2^4 trials. Consequently, the whole algorithm is probabilistic polynomial time. Our implementation using GNU MP finds preimages in less than 2 minutes for 1024-bit parameters on an Intel Pentium M processor 1.73GHz.

5 Collisions for the Morgenstern Hash Function

Now we show how to adapt Zémor and Tillich's algorithm for finding collisions in Morgenstern hashes when $q = 2$. Our algorithm lifts the representation problem from $SL(2, \mathbb{F}_{2^n})$ to a subset Ω of $SL(2, \mathbb{A})$ where $\mathbb{A} = \mathbb{F}_2[x, y]/(y^2 + y + 1)$ (in this section, i will denote a root of $i^2 + i + 1 = 0$ in $\mathbb{A}$ while $\mathbf{i}$ is a root of the same equation in $\mathbb{F}_{2^n}$). The relevant set is

$$\Omega = \left\{ \begin{pmatrix} a + bi & c + di \\ x(c + di + d) & a + bi + b \end{pmatrix} | (a, b, c, d) \in E_e \text{ for some integer } e > 0 \right\}$$

where E_e is the set of 4-tuples $(a, b, c, d) \in \mathbb{F}_2[x]$ such that

$$\begin{cases} (a^2 + b^2 + ab) + (c^2 + d^2 + cd)x = (1 + x)^e \\ a \equiv 1 \bmod x \\ b \equiv 0 \bmod x. \end{cases}$$

Again call the first of these equations the *norm equation*. We point out that in this section, small letters a, b, c, d, i, p are polynomials in x over $\mathbb{F}_2$, while capitalized letters will be used for elements of the field $\mathbb{F}_{2^n}$. By [9], corollary 5.4 and 5.7, if we restrict E_e to tuples (a, b, c, d) not divisible by $(1+x)$, the elements of Ω have a unique factorization in terms of the lifts of the graph generators:

$$g_0' = \begin{pmatrix} 1 & 1 + i \\ ix & 1 \end{pmatrix}, \quad g_1' = \begin{pmatrix} 1 & 1 \\ x & 1 \end{pmatrix}, \quad g_2' = \begin{pmatrix} 1 & i \\ (1 + i)x & 1 \end{pmatrix}.$$

Moreover, the "reduction modulo p" $(a,b,c,d) \to (A,B,C,D) = (a,b,c,d) \bmod p$ gives a homomorphism from Ω to $SL(2,\mathbb{F}_{2^n})$:

$$\begin{pmatrix} a+bi & c+di \\ x(c+di+d) & a+bi+b \end{pmatrix} \to \begin{pmatrix} A+B\mathbf{i} & C+D\mathbf{i} \\ x(C+D\mathbf{i}+D) & A+B\mathbf{i}+B \end{pmatrix}.$$

From this it is now clear how the second and third steps of Zémor and Tillich algorithm will work for Morgenstern hashes, so we now give details for first step. This amounts to lifting the representation problem, that is finding $a,b,c,d,\lambda \in \mathbb{F}_{2^n}$ satisfying the following conditions:

$$\begin{cases} (a,b,c,d) \in E_e \\ (a,b,c,d) \quad \text{not divisible by x+1} \\ (a,b,c,d) \equiv \lambda(1,0,0,0) \bmod p. \end{cases}$$

Write $b = xpb'$, $c = pc'$, $d = pd'$ for $b', c', d' \in \mathbb{F}_2[x]$ and arbitrarily choose $e = 2k$ and $a = (1+x)^k + xpm$, with $k \in \mathbb{Z}$ and $m \in \mathbb{F}_2[x]$ still to be determined. Note that such an a satisfies $a \equiv 1 \bmod x$. The norm equation becomes

$$x^2p^2m^2 + x^2p^2b'^2 + xpb'\left((1+x)^k + mxp\right) + xp^2(c'^2 + d'^2 + c'd') = 0.$$

Simplifying by xp we get

$$xpm^2 + xpb'^2 + b'\left((1+x)^k + xpm\right) + p(c'^2 + d'^2 + c'd') = 0.$$

Reducing this equation modulo p we get $b'\left((1+x)^k + xpm\right) \equiv 0$ which implies $b' = pb''$ for some $b'' \in \mathbb{F}_p$. The norm equation becomes

$$xpm^2 + xp^3b''^2 + pb''\left((1+x)^k + xpm\right) + p(c'^2 + d'^2 + c'd') = 0.$$

Simplifying again by p we get

$$c'^2 + d'^2 + c'd' = n(b'', m, k) := xm^2 + xp^2b''^2 + b''(1+x)^k + b''xpm.$$

Our approach for step 1 will be to generate random m and b'' (with $x+1 \nmid b''$) until the equation $c'^2 + d'^2 + c'd' = n(b'', m, k)$ has solutions, then to solve this equation for c', d'. As will be clear later, the equation has a solution if and only if all the irreducible factors of n are of even degree. So in particular

- We will choose $b'' = b^{(3)}x + 1$ for some $b^{(3)} \in \mathbb{F}_2[x]$ to avoid an x factor.
- As the term $xp^2b''^2$ is of odd degree, we will make another term of higher even degree, with the following strategy:
  - Choose b'' and m randomly of degree equal to or less than R.
  - Choose $k = 2\deg(p) + \deg(b'') + 2 + (\deg(b'') + \epsilon)$ where $\epsilon = 0$ if $\deg(b'')$ is even and $\epsilon = 1$ if $\deg(b'')$ is odd.

If R is large enough we get an n with the desired property after sufficiently many random trials on b'' and m. In our implementation, we chose $R = 10$ which is more than enough for 1024-bit parameters. It remains to show how to solve the equation $c'^2 + d'^2 + c'd' = n$ and to explain the condition on the degrees of irreducible factors of n. We begin with the solution of the equation.

Solving $c^2 + d^2 + cd = n$. It is enough to have an algorithm solving it when n is irreducible. Indeed, if $c_1^2 + d_1^2 + c_1 d_1 = n_1$ and $c_2^2 + d_2^2 + c_2 d_2 = n_2$ then $(c_3, d_3) = (c_1 c_2 + d_1 d_2, c_1 d_2 + c_2 d_1 + d_1 d_2)$ satisfies $c_3^2 + d_3^2 + c_3 d_3 = n_1 n_2$. So suppose n is irreducible of even degree.

We describe a continued fraction algorithm for polynomials over $\mathbb{F}_2$ and then we use it to solve the equation. For a fraction $\xi = \frac{P}{Q}$ where P and Q are polynomials, let $P = a_0 Q + r_0$ where $\deg r_0 < \deg Q$. Let $Q = a_1 r_0 + r_1$ with $\deg r_1 < \deg r_0$, then recursively for $i = 2, ...,$ define $r_{i-2} = a_i r_{i-1} + r_i$ with $\deg r_i < \deg r_{i-1}$. (This is the Euclidean algorithm applied to the ring $\mathbb{F}_2[x]$). Define $p_0 = a_0$, $q_0 = 1$, $p_1 = a_0 a_1 + 1$, $q_1 = a_1$, and then recursively $p_i = a_i p_{i-1} + p_{i-2}$ and $q_i = a_i q_{i-1} + q_{i-2}$. (The fraction p_i/q_i is the i^{th} truncated continued fraction of P/Q.) We see recursively that $q_i p_{i-1} + q_{i-1} p_i = 1$, so $\frac{p_i}{q_i} + \frac{p_{i-1}}{q_{i-1}} = \frac{1}{q_{i-1} q_i}$ and

$$\frac{P}{Q} = a_0 + \sum_{i=0}^{n} \frac{1}{q_{i+1} q_i}$$

where n is the first i such that $p_i/q_i = P/Q$. Define a "norm" v on quotients of polynomials as follows: $v\left(\frac{a}{b}\right) = \deg a - \deg b$ if $a, b \neq 0$, $v\left(\frac{a}{b}\right) = 0$ if $b = 0$, and $v\left(\frac{a}{b}\right) = -\infty$ if $a = 0$. Note that $v(q_{i+1}) \geq v(q_i)$, $v(p_{i+1}) \geq v(p_i)$, and that

$$v\left(\frac{P}{Q} + a_0 + \sum_{i=0}^{n'-1} \frac{1}{q_{i+1} q_i}\right) = v\left(\sum_{i=n'}^{n} \frac{1}{q_{i+1} q_i}\right) \leq -v(q_{n'+1}) - v(q_{n'})$$

As n has even degree, we can compute $\alpha \in \mathbb{F}_2[x]$ such that $\alpha^2 + \alpha + 1 \equiv 0 \bmod n$ (see next paragraph). We apply a continued fraction expansion to $\xi = \frac{\alpha}{n}$ and let p_i/q_i be the successive approximations. Let j be such that

$$v(q_j) \leq \frac{v(n)}{2} \leq v(q_{j+1}).$$

We have

$$q_j^2 + (q_j \alpha + p_j n)^2 + q_j (q_j \alpha + p_j n) \equiv q_j^2 + q_j^2 \alpha^2 + q_j^2 \alpha \equiv 0 \bmod n.$$

On the other hand, as

$$\begin{aligned} \deg(q_j \alpha + p_j n) &= v(n) + v(q_j) + v\left(\xi + \frac{p_j}{q_j}\right) \leq v(n) + v(q_j) - v(q_j) - v(q_{j+1}) \\ &\leq v(n)/2 = \deg(n)/2 \end{aligned}$$

we have

$$v\left(q_j^2 + (q_j \alpha + p_i n)^2 + q_i (q_j \alpha + p_j n)\right) \leq 2 \max\left(\deg(q_j), \deg(q_j \alpha + p_j n)\right) \leq \deg n.$$

Consequently,

$$q_j^2 + (q_j \alpha + p_j n)^2 + q_j (q_j \alpha + p_j n) = n$$

and $(c, d) = (q_j, q_j \alpha + p_j n)$ is a solution to $c^2 + d^2 + cd = n$.

Solutions to $\alpha^2 + \alpha + 1 \equiv 0 \bmod n$. As the map $x \to x^2 + x$ is linear in $\mathbb{F}_2$, solutions to this equation, if there are any, are found easily by writing down then solving a linear system of equations. We conclude the exposition of our algorithm by showing the following lemma.

Lemma 2. *For n irreducible, $\alpha^2 + \alpha + 1 \equiv 0 \bmod n$ has solutions if and only if $d := \deg(n)$ is even.*

($\Rightarrow$) Suppose α satisfies $\alpha^2 + \alpha + 1 \equiv 0 \bmod n$. Then $1 = \alpha + \alpha^2$. Squaring each side we get $1 = \alpha^2 + \alpha^{2^2}$, then squaring again and again we get $1 = \alpha^{2^2} + \alpha^{2^3}$,... until $1 = \alpha^{2^d} + \alpha^{2^{d-1}} = \alpha + \alpha^{2^{d-1}}$. Summing up these equations we get $d = 0$, so d must be even.

($\Leftarrow$) Now suppose d is even. Let β be a generator of $\mathbb{F}_{2^d}^*$ and let $\alpha = \beta^{\frac{2^d-1}{3}}$. Then $\alpha^3 = 1$ and $\alpha \neq 1$ so $\alpha^2 + \alpha + 1 = 0$. □

Runtime analysis. We give some estimates for the complexity of our algorithm. Assuming the polynomial n generated from random (b'', m) behaves like random polynomials of degree k, the number of its irreducible factors is asymptotically $K = \mathcal{O}(\log \deg n)$ [5]. For n of degree even, we can reasonably approximate the probability that all its factors are of even degree by $(1/2)^K$, hence we will need $2^K = \mathcal{O}(\log n) = \mathcal{O}(\deg p)$ random trials. The factorization of n can be done in $\mathcal{O}(\log^{2+\epsilon} n)$ [12] and the continued fraction algorithm is of complexity $\mathcal{O}(\deg n)$, so the global complexity of our algorithm is probabilistic polynomial time in $\deg p$. Our implementation of the algorithm finds collisions for 1024-bit parameters in a few seconds on a Pentium Intel M processor 1.73GHz.

6 Discussion and Further Work

In this paper, we presented efficient algorithms finding preimages for the LPS hash function and collisions for the Morgenstern hash function with $q = 2$. Similar algorithms with the same complexity can be derived for finding preimages for the Morgenstern hash function and for different q values. Our algorithms build upon the Zémor and Tillich algorithm [17] although they are not trivial extensions of it.

The modified version of LPS hashes proposed by Zémor and Tillich remains unbroken so far regarding both the collision and the preimage properties, as well as their original scheme [13] (with carefully chosen parameters) that used as graph generators $A_0 = \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix}$ and $A_1 = \begin{pmatrix} x & x+1 \\ 1 & 1 \end{pmatrix}$. To avoid our new attack, we suggest modifying the Morgenstern hash function as follows: multiply by $g_0 g_1$ if the bit is 0 and by $g_0 g_2$ if the bit is 1. However, it is not clear what would be the advantages of such a scheme compared to Zémor and Tillich's, as it would not necessarily have better expansion properties, and comparing the graph generators, it will certainly be slower.

In further work, we would like to study the applicability of our algorithm to the Zémor-Tillich (ZT) hash function. The Cayley graphs used in ZT hashes can be naturally embedded into Morgenstern graphs, so our cryptanalysis of

Morgenstern hashes might actually open new perspectives on breaking the ZT scheme. Our results may also have applications outside the cryptographic community. The preimage finding algorithm actually solves the diophantine equation (1) which at first sight seems to be a very hard problem. Our path-finding and Zémor and Tillich cycle-finding may improve understanding of LPS graphs when considering their Ihara Zeta-function. Finally, expander graphs have numerous applications in computer science [6], some of which could benefit from our new path-finding algorithm.

Because of all these actual and potential applications, we stress that our algorithms and their running time estimates still may and should be improved in many ways. The algorithm of Section 4 gives paths of length about $8 \log p$ while the diameter of LPS graphs is known to be $2 \log p$. Choosing a smaller k value in the algorithm will decrease this length and may also improve the running time. Finding other decompositions with less than 4 diagonal matrices is another interesting approach. Finally, adapting our algorithms to make them deterministic is a particularly interesting open problem.

References

1. http://csrc.nist.gov/groups/ST/hash/documents/SHA-3_FR_Notice_Nov02_2007%20-%20more%20readable%20version.pdf
2. FIPS 180-2 secure hash standard
3. Charles, D.X., Goren, E.Z., Lauter, K.E.: Cryptographic hash functions from expander graphs. Journal of Cryptology (to appear)
4. Contini, S., Lenstra, A.K., Steinfeld, R.: VSH, an efficient and provable collision-resistant hash function. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 165–182. Springer, Heidelberg (2006)
5. Flajolet, P., Soria, M.: Gaussian limiting distributions for the number of components in combinatorial structures. J. Comb. Theory Ser. A 53(2), 165–182 (1990)
6. Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bull. Amer. Math. Soc. 43, 439–561 (2006)
7. Lubotzky, A., Phillips, R., Sarnak, P.: Ramanujan graphs. Combinatorica 8, 261–277 (1988)
8. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: Provably secure FFT hashing. In: NIST 2nd Cryptogaphic Hash Workshop (2006)
9. Morgenstern, M.: Existence and explicit construction of $q + 1$ regular Ramanujan graphs for every prime power q. Journal of Combinatorial Theory B 62, 44–62 (1994)
10. Petit, C., Lauter, K.E., Quisquater, J.-J.: Full cryptanalysis of LPS and Morgenstern hash functions. Cryptology ePrint Archive, Report 2008/173 (2008), http://eprint.iacr.org/
11. Petit, C., Lauter, K.E., Quisquater, J.-J.: Cayley hashes: A class of efficient graph-based hash functions (preprint, 2007)
12. Shoup, V.: On the deterministic complexity of factoring polynomials over finite fields. Inf. Process. Lett. 33(5), 261–267 (1990)
13. Tillich, J.-P., Zémor, G.: Hashing with SL_2. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 40–49. Springer, Heidelberg (1994)
14. Tillich, J.-P., Zémor, G.: Group-theoretic hash functions. In: Cohen, G., Lobstein, A., Zémor, G., Litsyn, S.N. (eds.) Algebraic Coding 1993. LNCS, vol. 781, pp. 90–110. Springer, Heidelberg (1994)

15. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
16. Zémor, G.: Hash functions and Cayley graphs. Des. Codes Cryptography 4(4), 381–394 (1994)
17. Zémor, G., Tillich, J.-P.: Collisions for the LPS expander graph hash function. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965. Springer, Heidelberg (2008)

A Toy Example of the Preimage-Finding (Path-Finding) Algorithm in the LPS Graph

As an example of our preimage algorithm, we now give a second preimage for the message $m =$"This is not for NIST", when the parameters are $p = 1125899906842769$ and $l = 5$. The ASCII code for m is 01010100 01101000 01101001 01110011 00100000 01101001 01110011 00100000 01101110 01101111 01110100 00100000 01100110 01101111 01110010 00100000 01001110 01001001 01010011 01010100 which in base 5 gives 3023231443000032312104001244030134 2104032442012221213343131044243202 1. We start at the identity, with g_{IV} the identity and $g_0 = M_1$. We identify the six graph generators

$$M_{\pm 1} = \begin{pmatrix} 1 \pm 2\mathbf{i} & 0 \\ 0 & 1 \mp 2\mathbf{i} \end{pmatrix}, \quad M_{\pm 2} = \begin{pmatrix} 1 & \pm 2 \\ \mp 2 & 1 \end{pmatrix} \quad M_{\pm 3} = \begin{pmatrix} 1 & 2\mathbf{i} \\ 2\mathbf{i} & 1 \end{pmatrix}$$

with their indices. The function π we choose is given in figure A. The hash value obtained is

$$M = \begin{pmatrix} 1113908155375639 & 815055784352014 \\ 485525153198538 & 30164330826615 \end{pmatrix}.$$

	-3	-2	-1	1	2	3
0	-3	3	2	1	-1	-2
1	2	-3	3	2	1	-1
2	-1	-2	-3	3	2	1
3	1	-1	-2	-3	3	2
4	2	1	-1	-2	-3	3

Fig. 1. Table for the function π: the table gives the index of the next matrix for a given current matrix and a given base 5 digit

We apply our path-finding algorithm on M. First, we get a matrix decomposition as in Section 4. After 11 trials, the resulting λ, α, ω, β_1 and β_2 values are

$$\lambda = 1051846637406052$$
$$\alpha = 698130975272599$$
$$\omega = 846326642296745$$
$$\beta_1 = 150389273084944$$
$$\beta_2 = 480539407839455.$$

Then we factorize

$$M_\alpha := \begin{pmatrix} 1 & 0 \\ 0 & \alpha \end{pmatrix} = \begin{pmatrix} 349065487636300 + 795285597612250\mathbf{i} & 0 \\ 0 & 349065487636300 - 795285597612250\mathbf{i} \end{pmatrix}.$$

We choose $k = 48$, resulting in $\lambda = 222458048101540$ and $m = 1121038768144160066886982393688699301560731956564062 5$. After 234 random trials for x, we finally get $x = 523712450310834$, $w = 207632734870715$, and $n = 4.2489205976128525372183128649803320961$. The Euclidean algorithm gives us the solution $y = 2782001231666122912$, $z = 1489057773063985790$. So the lift of M_α is

$$M'_\alpha = \left(\begin{array}{c|c} 311426103887630914544037511835 & 313225492756935640601527301242 3328 \\ +\mathbf{i}76656548074545418488716312 4346 & +\mathbf{i}16765300079762426632936979802525 10 \\ \hline -313225492756935640601527301242 3328 & 311426103887630914544037511835 \\ +\mathbf{i}16765300079762426632936979802525 10 & -\mathbf{i}76656548074545418488716312 4346 \end{array}\right).$$

We multiply M'_α by each of the lifts of the graph generators. Since $M'_\alpha g'_3$ is divisible by $l = 5$, g'_{-3} is the last (right-hand) factor of M'_α. After $2k$ steps, we get the whole factorization of M'_α, which we translate into a factorization of M_α whose indices are 3 -1 2 2 3 1 1 3 1 3 3 3 2 2 3 -1 2 1 1 -3 1 1 1 3 -1 2 -3 2 3 1 -2 -2 -2 1 2 1 1 -3 2 1 2 1 -2 3 -1 3 2 -3 -2 3 1 -2 3 3 2 -3 -1 2 2 2 -1 -3 -1 -3 2 3 1 2 -3 -1 3 2 2 1 3 -2 -3 1 3 -2 -1 -2 3 1 3 2 1 -2 -1 -1 -3 2 1 1 -2 -3. We get the factorizations of M_ω, M_{β_1} and M_{β_2} the same way. Finally, we put all the pieces of information together and get the sequence -3 -2 1 1 2 -3 -1 -1 -2 1 2 3 1 3 -2 -1 -2 3 1 -3 -2 3 1 2 2 3 -1 -3 2 1 3 2 - 3 -1 -3 -1 2 2 2 -1 -3 2 3 3 -2 1 3 -2 -3 2 3 -1 3 -2 1 2 1 2 -3 1 1 2 1 -2 -2 - 2 1 3 2 -3 2 -1 3 1 1 1 -3 1 1 2 -1 3 2 2 3 3 3 1 3 1 1 3 2 2 -1 3 2 3 -1 -3 -2 -2 1 3 2 -3 -3 2 3 -2 -3 -3 -2 -1 -2 3 1 1 2 3 2 1 1 -3 1 2 2 1 -2 1 1 2 -3 -2 3 -2 -3 1 1 3 1 2 1 -3 -1 -3 -3 -1 2 3 1 -3 -3 -1 -1 2 -1 3 -2 1 -3 -3 -1 -2 1 1 -2 -1 -1 3 -2 3 2 2 1 -3 -2 -1 -3 -1 -3 -1 3 2 -1 3 3 -2 -1 -2 1 1 2 2 -3 -1 3 2 2 -3 -2 -3 -1 -3 1 -3 -2 -1 3 1 -2 3 -2 3 2 1 3 -2 -2 -3 -3 -2 -2 3 -2 -2 -3 -1 3 1 3 -1 -3 -3 -3 -3 -2 1 3 3 1 -2 3 -1 -2 1 2 -3 1 -2 -2 1 -2 -2 -1 2 2 2 2 2 1 -3 1 1 2 1 1 3 3 -1 3 3 -2 -1 3 1 2 -1 2 3 -1 2 -3 -2 1 -2 1 1 3 -2 2 -2 -3 -2 -1 3 3 -2 -1 3 2 -3 2 3 -2 1 1 1 1 -2 1 1 2 -1 3 -1 -2 -1 -2 -1 -2 -2 3 2 1 3 - 2 -2 -3 -3 -2 3 3 2 1 1 1 1 3 2 -1 -3 2 -3 -2 -1 -3 1 -2 -2 -2 1 -2 -1 -2 -1 2 - 1 -3 -1 -3 -2 -1 -2 -3 -1 -3 -1 2 2 3 3 -2 -2 -2 -3 -1 -1 -1 2 -3 -1 3 -1 2 -3 - 3 that collides with the original message "This is not for NIST".

B Collisions for Morgenstern Hashes, $q = 2$ and $\deg p(x) = 20$

Now we give a small example for our collision-finding algorithm. The polynomial we choose to target is $p(x) = x^{20} + x^{17} + x^{14} + x^{13} + x^{12} + x^{11} + x^9 + x^7 + x^5 + x^3 + x^2 + x + 1$. We choose $R = 10$ and generate random m and b''. After 3 random trials we get $m = x^9 + x^8 + x^7 + x^6 + x^5 + x^4$, $b'' = x^{10} + x^8 + x^5 + x^2 + 1$ so $k = 52$, $a = x^{52} + x^{48} + x^{36} + x^{32} + x^{30} + x^{25} + x^{24} + x^{22} + x^{20} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^9 + x^4 + x^3 + x + 1$, $b = x^{51} + x^{50} + x^{48} + x^{47} + x^{46} + x^{45} + x^{44} + x^{40} + x^{39} + x^{38} + x^{37} + x^{36} + x^{35} + x^{34} + x^{32} + x^{31} + x^{30} + x^{29} + x^{28} + x^{27} + x^{25} + x^{24} + x^{23} +$

$x^{22}+x^{20}+x^{19}+x^{18}+x^{16}+x^{11}+x^{10}+x^9+x^8+x^5+x^4+x^3+x^2+x$ and $n = x^{62}+x^{61}+x^{59}+x^{57}+x^{55}+x^{53}+x^{52}+x^{51}+x^{50}+x^{49}+x^{48}+x^{46}+x^{45}+x^{40}+x^{37}+x^{31}+x^{29}+x^{28}+x^{26}+x^{25}+x^{24}+x^{23}+x^{16}+x^{15}+x^{13}+x^{12}+x^{10}+x^6+x^5+x^3+1$.

The polynomial n has three factors $n_1 = x^{56}+x^{54}+x^{53}+x^{50}+x^{48}+x^{46}+x^{44}+x^{40}+x^{36}+x^{34}+x^{33}+x^{30}+x^{29}+x^{22}+x^{20}+x^{18}+x^{13}+x^{11}+x^7+x^6+x^5+x^3+1$, $n_2 = x^4+x^3+x^2+x+1$ and $n_3 = x^2+x+1$ which are all of even degrees. For each factor n_i we compute α such that $\alpha^2+\alpha+1 \equiv 0 \bmod n_i$ and use this value and the continued fraction algorithm to recover (c_i, d_i) such that $c_i^2+d_i^2+c_i d_i \equiv 0 \bmod n_i$: we get $(c_1, d_1) = (x^{26}+x^{25}+x^{24}+x^{21}+x^{20}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^8+x^6+x^5+x+1, x^{28}+x^{23}+x^{21}+x^{19}+x^{15}+x^{13}+x^{10}+x^7+x^5+x^4+x^2+x+1)$, $(c_2, d_2) = (x, x^2+1)$ and $(c_3, d_3) = (x, 1)$.

Combining these partial results we get $c = x^{51}+x^{50}+x^{47}+x^{41}+x^{40}+x^{36}+x^{31}+x^{27}+x^{26}+x^{25}+x^{24}+x^{23}+x^{22}+x^{20}+x^{18}+x^{17}+x^{16}+x^{14}+x^{12}+x^{11}+x^{10}+x^9+x^7+x^4$ and $d = x^{51}+x^{50}+x^{49}+x^{48}+x^{47}+x^{45}+x^{44}+x^{43}+x^{42}+x^{39}+x^{36}+x^{33}+x^{31}+x^{30}+x^{29}+x^{27}+x^{26}+x^{25}+x^{22}+x^{21}+x^{20}+x^{18}+x^{17}+x^{16}+x^{14}+x^{13}+x^9+x^7+1$.

We can verify that

$$(a^2+b^2+ab)+(c^2+d^2+cd)x = (1+x)^{2k}$$

and $(a, b, c, d) \equiv (1+x)^k(1, 0, 0, 0) \bmod p$. We factorize the lifted matrix and, using the indices of the generators given in Section 5, we get the following collision with the void message: 0 2 0 2 0 1 2 1 2 1 2 1 2 0 2 0 2 0 2 0 2 0 1 0 2 0 2 1 0 2 1 0 1 0 2 1 0 1 2 1 2 1 2 1 0 2 1 0 1 2 0 1 0 1 0 1 0 2 1 0 1 2 0 2 1 2 0 2 0 1 2 0 2 0 1 0 2 1 2 1 0 2 0 1 0 1 2 0 2 0 2 0 2 0 1 2 1 0 2 0 2 1 0 1.

A New DPA Countermeasure Based on Permutation Tables

Jean-Sébastien Coron

University of Luxembourg
jean-sebastien.coron@uni.lu

Abstract. We propose and analyse a new countermeasure against Differential Power Analysis (DPA) for the AES encryption algorithm, based on permutation tables. As opposed to existing AES countermeasures, it does not use random masking. We prove that our new countermeasure is resistant against first-order DPA; we also show that it is quite efficient in practice.

1 Introduction

The AES [4] encryption algorithm is with DES the most widely used encryption algorithm. However, it is easy to see that without modification, AES is vulnerable to Differential Power Analysis as introduced by Kocher *et al.* in [6,7]. A DPA attack consists in extracting information about the secret key of a cryptographic algorithm, by studying the power consumption of the electronic device during the execution of the algorithm. The attack was first described on the DES encryption scheme, but it was soon understood that the attack could easily be extended to other symmetrical cryptosystems such as the AES, and also to public-key cryptosystems such as RSA and Elliptic-Curves Cryptosystems [3].

A common technique to protect secret-key algorithms against side-channel attacks consists in masking all data with a random integer, as suggested in [2]. The masked data and the random integer are then processed separately and eventually recombined at the end of the algorithm. An attacker trying to analyse power consumption at a single point will obtain only random values; therefore, the implementation will be secure against first-order Differential Power Analysis (DPA). In order to obtain valuable information about the key, the attacker must correlate the power consumption at multiple points during the execution of the algorithm; this is called a High Order Differential Power Analysis (HO-DPA); such attack usually requires a much larger number of power consumption curves, which makes it infeasible in practice if the number of executions is limited (for example, by using a counter). Many AES countermeasures have been described based on random masking [1,5,9,10,12].

In this article we propose a different countermeasure against DPA for AES, based on permutation tables. The main difference with existing AES countermeasures is that it avoids random masking; in practice this can be an advantage because random masking is subject to numerous patents [7]. We prove that our

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 278–292, 2008.

countermeasure is resistant against first-order DPA (like the random masking countermeasure) and we show that its efficiency is comparable to that of the random masking countermeasure.

It works as follows: at initialisation time a randomised permutation table is generated in RAM; this permutation table is then applied to the message and to the key; then all intermediate variables that appear during the course of the algorithm remain in permuted form; eventually the inverse permutation is applied to obtain the resulting ciphertext.

We also describe a technique to reduce the RAM consumption of our permutation table countermeasure, at the cost of increasing the running time. Our technique is based on a compression scheme proposed in [11] for the classical random masking countermeasure; here we adapt this scheme to permutation tables. We show that this variant is also secure against first-order DPA. Finally, we also provide the result of implementations that show that our countermeasure is reasonably efficient in practice, as it is only roughly four times slower than the classical masking countermeasure, for a comparable RAM requirement (see Table 1 in Section 7 for a detailed comparison).

2 The AES Encryption Algorithm

In this section we recall the main operations involved in the AES algorithm. We refer to [4] for a full description. AES operates on a 4×4 array of bytes $s_{i,j}$, termed the state. For encryption, each AES round (except the last) consists of four stages:

1. **AddRoundKey:** each byte of the state is xored with the round key $k_{i,j}$, derived from the key schedule:
$$s_{i,j} \leftarrow s_{i,j} \oplus k_{i,j}$$
2. **SubBytes:** each byte of the state is updated using an 8-bit S-box:
$$s_{i,j} \leftarrow S(s_{i,j})$$
3. **ShiftRows:** the bytes of the state are cyclically shifted in each row by a certain offset; the first row is left unchanged.
4. **MixColumns:** the bytes of the state are modified column by column as follows:
$$\begin{aligned} s'_{0,c} &\leftarrow (\mathsf{02} \cdot s_{0,c}) \oplus (\mathsf{03} \cdot s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &\leftarrow s_{0,c} \oplus (\mathsf{02} \cdot s_{1,c}) \oplus (\mathsf{03} \cdot s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &\leftarrow s_{0,c} \oplus s_{1,c} \oplus (\mathsf{02} \cdot s_{2,c}) \oplus (\mathsf{03} \cdot s_{3,c}) \\ s'_{3,c} &\leftarrow (\mathsf{03} \cdot s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\mathsf{02} \cdot s_{3,c}) \end{aligned}$$

The pseudo-code for AES encryption with a 128-bit key is recalled in Figure 1 in Appendix. The word array w contains the round keys that are generated by the key-schedule algorithm. We refer to [4] for a more detailed description.

For decryption, each round (except the last) consists in the following operations:

1. **InvShiftRows:** is the inverse of the ShiftRows operation. The bytes of the state are cyclically shifted in each row by a certain offset; the first row is left unchanged.
2. **InvSubBytes:** is the inverse of the SubBytes operation. The inverse S-box S^{-1} is applied on each byte of the state.
3. **AddRoundKey:** the operation is equal to its own inverse.
4. **InvMixColumns:** is the inverse of the MixColumns operation. The bytes of the state are modified column by column as follows:

$$\begin{aligned}
s'_{0,c} &\leftarrow (\mathtt{0e} \cdot s_{0,c}) \oplus (\mathtt{0b} \cdot s_{1,c}) \oplus (\mathtt{0d} \cdot s_{2,c}) \oplus (\mathtt{09} \cdot s_{3,c}) \\
s'_{1,c} &\leftarrow (\mathtt{09} \cdot s_{0,c}) \oplus (\mathtt{0e} \cdot s_{1,c}) \oplus (\mathtt{0b} \cdot s_{2,c}) \oplus (\mathtt{0d} \cdot s_{3,c}) \\
s'_{2,c} &\leftarrow (\mathtt{0d} \cdot s_{0,c}) \oplus (\mathtt{09} \cdot s_{1,c}) \oplus (\mathtt{0e} \cdot s_{2,c}) \oplus (\mathtt{0b} \cdot s_{3,c}) \\
s'_{3,c} &\leftarrow (\mathtt{0b} \cdot s_{0,c}) \oplus (\mathtt{0d} \cdot s_{1,c}) \oplus (\mathtt{09} \cdot s_{2,c}) \oplus (\mathtt{0e} \cdot s_{3,c})
\end{aligned}$$

The pseudo-code for the inverse cipher is recalled in Figure 2 in Appendix. Finally, the key-schedule is based on the following operations:

1. **SubWord:** takes a four-byte input word and applies the S-box S to each of the four bytes.
2. **RotWord:** takes a word $[a_0, a_1, a_2, a_3]$ as input and performs a cyclic permutation to return $[a_1, a_2, a_3, a_0]$.
3. **Xor with Rcon:** takes as input a 32-bits word and xor it with the round constant word array $\mathsf{Rcon}[\mathsf{i}] = [(\mathtt{02})^{i-1}, \mathtt{00}, \mathtt{00}, \mathtt{00}]$, for round $1 \leq i \leq 10$.

We refer to [4] for a full description of the key-schedule.

3 The Permutation Table Countermeasure

In this section we describe our basic countermeasure with permutation tables. A variant with a time-memory trade-off is described in Section 6.

Our countermeasure consists in using a randomised representation of the state variables, using two independent 4-bit permutation tables p_1 and p_2 that are freshly generated before each new execution of the algorithm. More precisely, every intermediate byte $x = x_h \| x_l$, where x_h is the high nibble and x_l is the low nibble, will be represented in the following form:

$$P(x) = p_2(x_h) \| p_1(x_l)$$

This permuted representation is then used throughout the execution of AES. Eventually the original representation is recovered at the end of the encryption algorithm, by applying the inverse permutation.

3.1 Generation of Permutation Tables p_1 and p_2

The 4-bit permutation tables p_1 and p_2 are generated for each new execution of the algorithm as follows. Let s_0 be a fixed 4-bit permutation table; one can take for example:

$$s_0 = [14, 6, 0, 5, 9, 1, 4, 15, 8, 10, 12, 2, 3, 13, 11, 7]$$

One defines a sequence of permutations s_i for $i \geq 0$ as follows:

$$s_{i+1}(x) = s_0\left(s_i(x) \oplus k_i\right)$$

where each $k_i \in \{0,1\}^4$ is randomly generated. The 4-bit permutation p_1 is then defined as $p_1 := s_n$ for some n (in practice, one can take $n = 4$). One applies the same procedure to generate the other table p_2 (with independently generated k_i's). Every intermediate byte $x = x_h \| x_l$ that appear in AES is then represented as:

$$P(x) = p_2(x_h) \| p_1(x_l)$$

Therefore, P is a 8-bit permutation; its storage requires 16 bytes of RAM. In the following we explain how to use this permuted representation throughout the AES operations, so that the intermediate data are never manipulated in clear.

3.2 AddRoundKey

Given $P(x)$ and $P(y)$ we explain how to compute $P(x \oplus y)$ without manipulating x and y directly (since otherwise this would give a straightforward DPA attack).

We define the following two 8-bit to 4-bit *xor-tables*; for all $x', y' \in \{0,1\}^4$:

$$\mathsf{XT}_4^1(x', y') := p_1(p_1^{-1}(x') \oplus p_1^{-1}(y'))$$
$$\mathsf{XT}_4^2(x', y') := p_2(p_2^{-1}(x') \oplus p_2^{-1}(y'))$$

Those two tables require a total of 256 bytes in memory. Then given $p_1(x)$ and $p_1(y)$ one can compute $p_1(x \oplus y)$ using:

$$\mathsf{XT}_4^1(p_1(x), p_1(y)) = p_1(x \oplus y)$$

for all $x, y \in \{0,1\}^4$. Similarly, we have:

$$\mathsf{XT}_4^2(p_2(x), p_2(y)) = p_2(x \oplus y)$$

Using those two tables we define the following function for all $x', y' \in \{0,1\}^8$, where $x' = x'_h \| x'_l$ and $y' = y'_h \| y'_l$:

$$\mathsf{XT}_8(x', y') = \mathsf{XT}_4^2(x'_h, y'_h) \| \mathsf{XT}_4^1(x'_l, y'_l)$$

Then given $P(x)$ and $P(y)$, one can compute $P(x \oplus y)$ as:

$$\mathsf{XT}_8(P(x), P(y)) = P(x \oplus y) \tag{1}$$

The AddRoundKey operation can then be implemented as:

$$s'_{i,j} \leftarrow \mathsf{XT}_8(s'_{i,j}, k'_{i,j})$$

where $s'_{i,j} = P(s_{i,j})$ and $k'_{i,j} = P(k_{i,j})$. It is therefore necessary to use the permuted representation for the round keys. We further describe how this is done by modifying the key-schedule operations (see sections 3.9, 3.10 and 3.11).

3.3 SubBytes

Let S be the AES SBOX. We define the following randomised permutation S':

$$S'(x) = P(S(P^{-1}(x)))$$

Given $P(x)$, the permuted representation of $S(x)$ is then computed as:

$$P(S(x)) = S'(P(x))$$

The SubBytes operation on the permuted state variables can then be computed using the table S' as follows:

$$s'_{i,j} \leftarrow S'(s'_{i,j})$$

The randomised table $S'(x)$ requires 256 bytes in RAM.

3.4 ShiftRows

No modification is required.

3.5 MixColumns

Using $\mathsf{03} \cdot x = (\mathsf{02} \cdot x) \oplus x$, the MixColumns operation can be written as follows (first line):

$$s'_{0,c} \leftarrow (\mathsf{02} \cdot s_{0,c}) \oplus (\mathsf{02} \cdot s_{1,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus s_{3,c}$$

Therefore, we must be able to compute $P(\mathsf{02} \cdot x)$ from $P(x)$. For any $x = x_h \| x_l \in \{0,1\}^8$, from $x = (0\|x_l) \oplus (x_h\|0)$ we have:

$$\mathsf{02} \cdot x = (\mathsf{02} \cdot (0\|x_l)) \oplus (\mathsf{02} \cdot (x_h\|0))$$

and using equation (1), we obtain:

$$P(\mathsf{02} \cdot x) = \mathsf{XT}_8\big(P(\mathsf{02} \cdot (0\|x_l)), P(\mathsf{02} \cdot (x_h\|0))\big) \tag{2}$$

Therefore, for all $x' \in \{0,1\}^8$ where $x' = x'_h \| x'_l$, we define the following two tables (4-bit to 8-bit):

$$\mathsf{ML}(x'_l) := P\Big(\mathsf{02} \cdot \big(\quad 0 \quad \| \; p_1^{-1}(x'_l)\big)\Big)$$
$$\mathsf{MH}(x'_h) := P\Big(\mathsf{02} \cdot \big(\; p_2^{-1}(x'_h) \; \| \quad 0 \quad\big)\Big)$$

which gives for any $x_l, x_h \in \{0,1\}^4$:

$$\begin{aligned} \mathsf{ML}(p_1(x_l)) &= P\big(\mathsf{02} \cdot (\ 0\ \|\ x_l\ )\big) \\ \mathsf{MH}(p_2(x_h)) &= P\big(\mathsf{02} \cdot (\ x_h\ \|\ 0\ )\big) \end{aligned} \tag{3}$$

Using equations (2) and (3), given $P(x) = p_2(x_h)\|p_1(x_l)$, we can therefore compute $P(\mathsf{02} \cdot x)$ as follows:

$$P(\mathsf{02} \cdot x) = \mathsf{XT}_8(\mathsf{ML}(p_1(x_l)), \mathsf{MH}(p_2(x_h)))$$

For simplicity, given $P(x) = x'_h\|x'_l$, we denote:

$$D_2(x'_h\|x'_l) = \mathsf{XT}_8\big(\mathsf{ML}(x'_l),\ \mathsf{MH}(x'_h)\big)$$

so that for any $x \in \{0,1\}^8$:

$$P(\mathsf{02} \cdot x) = D_2(P(x)) \tag{4}$$

The first line of the MixColumns operation with permuted data can therefore be written as:

$$s''_{0,c} \leftarrow \mathsf{XT}_8\left(D_2(s'_{0,c}), \mathsf{XT}_8\left(D_2(s'_{1,c}), \mathsf{XT}_8\left(s'_{1,c}, \mathsf{XT}_8(s'_{2,c}, s'_{3,c})\right)\right)\right)$$

and the other three lines are implemented analogously. The storage of the two ML and MH tables requires 32 bits of RAM.

3.6 InvShiftRows

The algorithm remains the same.

3.7 InvSubBytes

This is similar to the SubBytes algorithm: we define the following randomised permutation S'^{-1}:

$$S'^{-1}(x) = P(S^{-1}(P^{-1}(x)))$$

Therefore, the InvSubBytes operation on the permuted state variables is computed using table $S'^{-1}(x)$ as follows:

$$s'_{i,j} \leftarrow S'^{-1}(s'_{i,j})$$

Note that one can generate the randomised table $S'^{-1}(x)$ from $S(x)$ only, so that it is not necessary to store $S^{-1}(x)$ in ROM, using the fact that:

$$S'^{-1} = P \circ S^{-1} \circ P^{-1} = (P \circ S \circ P^{-1})^{-1}.$$

3.8 InvMixColumns

The first line of the InvMixColumns operation is as follows:

$$s'_{0,c} \leftarrow (\mathtt{0e} \cdot s_{0,c}) \oplus (\mathtt{0b} \cdot s_{1,c}) \oplus (\mathtt{0d} \cdot s_{2,c}) \oplus (\mathtt{09} \cdot s_{3,c})$$

We have the following relations in $\mathsf{GF}(2^8)$:

$$\mathtt{0b} = \mathtt{09} \oplus \mathtt{02}, \quad \mathtt{0d} = \mathtt{0c} \oplus \mathtt{01}, \quad \mathtt{0e} = \mathtt{0c} \oplus \mathtt{02} \tag{5}$$

Therefore only two tables for computing the multiplication by 09 and 0c are required, from which multiplication by 0b, 0d and 0e can be computed without additional tables. More precisely, for all $x \in \{0,1\}^4$, we build the following 4-bit to 8-bit tables:

$$\begin{aligned}
\mathsf{TL}(x'_l) &:= P\Big(\mathtt{09} \cdot \big(\quad 0 \quad \| \; p_1^{-1}(x'_l)\big)\Big)\\
\mathsf{TH}(x'_h) &:= P\Big(\mathtt{09} \cdot \big(\; p_2^{-1}(x'_h) \| \quad 0 \quad\big)\Big)\\
\mathsf{RL}(x'_l) &:= P\Big(\mathtt{0c} \cdot \big(\quad 0 \quad \| \; p_1^{-1}(x'_l)\big)\Big)\\
\mathsf{RH}(x'_h) &:= P\Big(\mathtt{0c} \cdot \big(\; p_2^{-1}(x'_h) \| \quad 0 \quad\big)\Big)
\end{aligned}$$

Storing those four tables requires 64 bytes in RAM. Then, as in section 3.5, writing $x' = P(x) = x'_h \| x'_l = p_2(x_h) \| p_1(x_l)$, we obtain:

$$\begin{aligned}
P(\mathtt{09} \cdot x) &= \mathsf{XT}_8\big(\mathsf{TH}(x'_h),\ \mathsf{TL}(x'_l)\big)\\
P(\mathtt{0c} \cdot x) &= \mathsf{XT}_8\big(\mathsf{RH}(x'_h),\ \mathsf{RL}(x'_l)\big)
\end{aligned}$$

As previously we denote:

$$\begin{aligned}
D_9(x'_h \| x'_l) &= \mathsf{XT}_8\big(\mathsf{TH}(x'_h),\ \mathsf{TL}(x'_l)\big)\\
D_c(x'_h \| x'_l) &= \mathsf{XT}_8\big(\mathsf{RH}(x'_h),\ \mathsf{RL}(x'_l)\big)
\end{aligned}$$

Using equations (5), we also denote:

$$\begin{aligned}
D_b(x) &= \mathsf{XT}_8(D_9(x), D_2(x))\\
D_d(x) &= \mathsf{XT}_8(D_c(x), x)\\
D_e(x) &= \mathsf{XT}_8(D_c(x), D_2(x))
\end{aligned}$$

The first line of the InvMixColumns operation can then be rewritten as follows:

$$s''_{0,c} \leftarrow \mathsf{XT}_8\left(D_e(s'_{0,c}), \mathsf{XT}_8\left(D_b(s'_{1,c}), \mathsf{XT}_8\left(D_d(s'_{2,c}), D_9(s'_{3,c})\right)\right)\right)$$

and the other three lines are rewritten analogously.

3.9 SubWord

The SubWord operation on the modified state variables is implemented like the SubByte operation.

3.10 RotWord

The RotWord operation remains unmodified.

3.11 Xor with Rcon

Let

$$R(i) = \mathtt{02}^{i-1}$$

for all $1 \leq i \leq 10$. We have $R(0) = \mathtt{01}$ and $R(i) = \mathtt{02} \cdot R(i-1)$ for all $1 \leq i \leq 10$. Therefore, letting $R'(i) := P(R(i))$, we have:

$$R'(i) = P(R(i)) = P(\mathtt{02} \cdot R(i-1)) = D_2(R(i-1))$$

Therefore the Rcon constant can be computed using the function $D_2(x)$ defined in section 3.5.

4 Security

In this section we show that our countermeasure in resistant against first-order DPA. This is due to the following lemma:

Lemma 1. *For a fixed key and input message, every intermediate byte that is computed in the course of the randomised AES algorithm has the uniform distribution in* $\{0,1\}^8$.

Proof. The proof follows directly from the fact that any intermediate AES data x is represented as $P(x)$, where $P(x_h \| x_l) = p_2(x_h) \| p_1(x_l)$ is the randomised permutation. From the construction of p_1, p_2, this implies that $P(x)$ is randomly distributed in $\{0,1\}^8$. □

The previous lemma implies that an attacker who makes statistics about power-consumption at a single point gets only random values; hence, the countermeasure is resistant against first-order DPA (like the random masking countermeasure). In order to obtain valuable information about the key, the attacker must correlate the power consumption at multiple points during the execution of the algorithm; this is called a High Order Differential Power Analysis (HO-DPA); such attack usually requires a much larger number of power consumption curves, which makes it infeasible in practice if the number of executions is limited (for example, by using a counter).

5 A Compression Scheme

A very nice compression scheme for SBOX tables has been proposed in [11]. This compression scheme works for SBOXes with a random mask; we recall it in this

section.[1] Then in Section 6 we show how to adapt it to our permutation table countermeasure.

Let $S(x)$ for $x \in \{0,1\}^8$ be a 8-bit to 8-bit SBOX. One defines $S_1(x)$ and $S_2(x)$ such that $S(x) = S_2(x)\|S_1(x)$ for all $x \in \{0,1\}^8$, where $S_1(x)$ and $S_2(x)$ are 4-bit values. Let $r_1, r_2 \in \{0,1\}^8$ and $s \in \{0,1\}^4$ be random masks, and let define the randomised table:

$$T(x) = S_1(x \oplus r_1) \oplus S_2(x \oplus r_2) \oplus s \tag{6}$$

which is a 8-bit to 4-bit table. Let $x' = x \oplus (r_1 \oplus r_2)$ be a masked data; we have from (6):

$$S_1(x) = S_1(x' \oplus r_1 \oplus r_2) = T(x' \oplus r_2) \oplus S_2(x') \oplus s$$
$$S_2(x) = S_2(x' \oplus r_1 \oplus r_2) = T(x' \oplus r_1) \oplus S_1(x') \oplus s$$

which gives:

$$S_1(x) \oplus s = T(x' \oplus r_2) \oplus S_2(x') \tag{7}$$
$$S_2(x) \oplus s = T(x' \oplus r_1) \oplus S_1(x') \tag{8}$$

Therefore given the masked data x' one can obtain a masked $S_1(x)$ and a masked $S_2(x)$, by using the randomised table T. The advantage is that the size of T is only half the size of a classically randomised SBOX table; here the storage of T requires only 128 bytes of RAM instead of 256 bytes for a classically randomised AES SBOX. More precisely, the algorithm is as follows:

InitTable(r_1, r_2, s)
1. Write $S(x) = S_2(x)\|S_1(x)$
2. Generate randomised table $T(x) = S_1(x \oplus r_1) \oplus S_2(x \oplus r_2) \oplus s$ for all $x \in \{0,1\}^8$

TableLookUp(x', r_1, r_2, s)
1. Generate a random $t \in \{0,1\}^4$
2. $u \leftarrow T(x' \oplus r_2) \oplus S_2(x')$
3. $v \leftarrow T(x' \oplus r_1) \oplus S_1(x') \oplus t$
4. Output $y = v\|u \in \{0,1\}^8$ and $r' = (s \oplus t)\|s$.

Here we add an additional masking step with t so that the values u and v are not masked with the same nibble s; from equations (7) and (8), we obtain that the value returned by TableLookUp is such that $y = S(x) \oplus r'$.

It is easy to see that this countermeasure is resistant against first-order DPA, as all the variables that appear in the TableLookUp function are uniformly distributed. Note that the tables S_1 and S_2 are only stored in ROM; they don't need to be randomised because in the TableLookUp algorithm those tables are accessed at point x' which is already randomised.

[1] In [11] the countermeasure is described in plain English which makes it difficult to understand; here we attempt to provide a more clear description.

It is also easy to see that the countermeasure can be generalised to any SBOX input and output size. Moreover, one can obtain a better compression factor by splitting $S(x)$ into more shares; for example, a 8-bit SBOX could be split into 8 tables (one for each output bit); then the resulting randomised T table would be 8 times smaller, at the cost of increasing the running time for every table look-up.

6 Time Memory Trade-Offs

In this section we describe three time-memory trade-offs. The goal is to reduce the RAM requirement of the permutation table countermeasure described in Section 3, for implementation on low-cost devices, at the cost of increasing the running time. The main time-memory tradeoff is based on the SBOX compression scheme recalled in the previous section. The second idea consists in using a single XOR table $\mathsf{XT}_4^1(x', y')$ instead of two XOR tables $\mathsf{XT}_4^1(x', y')$ and $\mathsf{XT}_4^2(x', y')$ as in Section 3.2. The third idea consists in removing tables TL, TH, RL and RH, by using a "Double-And-Add" approach. In Section 7 we describe the results of practical implementations of these time-memory trade-offs.

6.1 Compressed SBOX

The compression scheme of [11] recalled in the previous section was used for random masking; here we show how to adapt this compression scheme to our permutation table countermeasure.

We define a new permutation $P'(x) = p'_2(x_h) \| p'_1(x_l)$ where p'_1 and p'_2 are 4-bit permutations tables which are generated like p_1 and p_2. As previously, we write:

$$S(x) = S_2(x) \| S_1(x)$$

where $S_1(x)$ and $S_2(x)$ are 4-bit nibbles. We then define a randomised table:

$$T(x') = p_1\left(S_1\left(P^{-1}(x')\right)\right) \oplus p_2\left(S_2\left(P^{-1}\left(P'^{-1}(x')\right)\right)\right) \tag{9}$$

The randomised table $T(x')$ is a 8-bit to 4-bit table; therefore, its storage requires 128 bits in memory, instead of 256 bytes for the randomised table $S'(x')$ in Section 3.3. Writing $x' = P(x)$, we obtain from equation (9):

$$p_1(S_1(x)) = T(P(x)) \oplus p_2\left(S_2\left(P^{-1}\left(P'^{-1}\left(P(x)\right)\right)\right)\right)$$

This shows that given $P(x)$ we can compute $p_1(S_1(x))$ using randomised table T and table S_2 stored in ROM. Similarly, writing $x' = P'(P(x))$, we obtain from equation (9):

$$p_2(S_2(x)) = T\left(P'(P(x))\right) \oplus p_1\left(S_1\left(P^{-1}\left(P'\left(P(x)\right)\right)\right)\right)$$

This shows that given $P(x)$ we can compute $p_2(S_2(x))$ using randomised table T and table S_1 stored in ROM. Therefore, given $P(x)$ we can compute:

$$P(S(x)) = p_2\left(S_2(x)\right) \| p_1\left(S_1(x)\right)$$

using the following operations:

$\mathsf{InitTable}(P, P')$:

1. Write $S(x) = S_2(x) \| S_1(x)$
2. Generate table $T(x') = p_1\left(S_1\left(P^{-1}(x')\right)\right) \oplus p_2\left(S_2\left(P^{-1}\left(P'^{-1}(x')\right)\right)\right)$ for all $x' \in \{0,1\}^8$.

$\mathsf{TableLookUp}(x', T, P, P')$

1. $u \leftarrow T(x') \oplus p_2\left(S_2\left(P^{-1}\left(P'^{-1}(x')\right)\right)\right)$
2. $v \leftarrow T\left(P'(x')\right) \oplus p_1\left(S_1\left(P^{-1}\left(P'(x')\right)\right)\right)$
3. Output $v \| u$

Given the tables P, P', T and denoting $F(x') = \mathsf{TableLookUp}(x', T, P, P')$, the $\mathsf{SubBytes}$ operation on the permuted state variables is computed as follows:

$$s'_{i,j} \leftarrow F(s'_{i,j})$$

The table T requires 128 bytes in memory, and the additional permutations P' and P'^{-1} require 32 bytes in memory, so in total 160 bytes are required (instead of 256 bytes for the randomised table S'). The $\mathsf{InvSubBytes}$ operation on the permuted state variables with compressed inverse SBOX S^{-1} is obtained by replacing S by S^{-1} in the previous equations.

6.2 Single XOR Table

In Section 3.2 two 8-bit to 4-bit xor-tables are defined. In this section, we show that it is sufficient to define only one 8-bit to 4-bit xor-table. As in Section 3.2, we define:

$$\mathsf{XT}_4^1(x', y') := p_1(p_1^{-1}(x') \oplus p_1^{-1}(y'))$$

We also define the 4-bit to 4-bit permutations:

$$p_{12}(x) = p_1(p_2^{-1}(x)) \tag{10}$$
$$p_{21}(x) = p_2(p_1^{-1}(x)) \tag{11}$$

and we store those two permutation tables in RAM. Then we can define the function:

$$\mathsf{XT}_4^2(x', y') := p_{21}(\mathsf{XT}_4^1(p_{12}(x'), p_{12}(y')))$$

From equations (10) and (11) we obtain that for all $x_h, y_h \in \{0,1\}^4$:

$$\mathsf{XT}_4^2(p_2(x_h), p_2(y_h)) = p_{21}(\mathsf{XT}_4^1(p_1(x_h), p_1(y_h)) = p_{21}(p_1(x_h \oplus y_h)) = p_2(x_h \oplus y_h)$$

Therefore, the XT_4^2 function takes the same values as the XT_4^2 table defined in Section 3.2. The advantage is that only 16 bytes of RAM are necessary to store the permutation tables p_{12} and p_{21} instead of 128 bytes for the previous XT_4^2 table.

6.3 Double and Add for InvMixColumns

The first line of the InvMixColumns operation is as follows:

$$s'_{0,c} \leftarrow (\mathtt{0e} \cdot s_{0,c}) \oplus (\mathtt{0b} \cdot s_{1,c}) \oplus (\mathtt{0d} \cdot s_{2,c}) \oplus (\mathtt{09} \cdot s_{3,c})$$

In this section we show how to avoid the four tables TL, TH, RL and RH, using a Double and Add method. More precisely, using the existing D_2 function (see equation (4)) we define the following functions:

$$\begin{aligned}
D_4(x') &:= D_2(D_2(x')) \\
D_8(x') &:= D_2(D_4(x')) \\
D_9(x') &:= \mathsf{XT_8}(D_8(x'), x') \\
D_b(x') &:= \mathsf{XT_8}(D_9(x'), D_2(x')) \\
D_c(x') &:= \mathsf{XT_8}(D_8(x'), D_4(x')) \\
D_d(x') &:= \mathsf{XT_8}(D_c(x'), x') \\
D_e(x') &:= \mathsf{XT_8}(D_c(x'), D_2(x'))
\end{aligned}$$

Therefore no additional table is required beyond the already defined ML and MH tables. The first line of the InvMixColumns operation can then be rewritten as follows:

$$s''_{0,c} \leftarrow \mathsf{XT}_8\left(D_e(s'_{0,c}), \mathsf{XT}_8\left(D_b(s'_{1,c}), \mathsf{XT}_8\left(D_d(s'_{2,c}), D_9(s'_{3,c})\right)\right)\right)$$

and the other three lines are rewritten analogously.

6.4 Security

As for our basic permutation table countermeasure, all the intermediate variables that appear in the time-memory tradeoff variants have the uniform distribution:

Lemma 2. *For a fixed key and input message, every intermediate byte that is computed in the course of the randomised AES algorithm with any of the three time-memory trade-offs has the uniform distribution in* $\{0,1\}^8$.

Therefore, the three previous time-memory tradeoffs are secure against first-order DPA.

7 Implementation

We summarise in Table 1 the timings observed and RAM needed for each AES operation. The timings are based on an implementation in C on a 2 GHz laptop under Linux. The RAM requirements are based on Tables 2 and 3 in Appendix. These timings show that the new countermeasure is reasonably efficient, as it is only roughly four times slower than AES with masking countermeasure, for a comparable amount of memory. We note that the time-memory tradeoffs enable to spare 272 bytes in RAM compared to our basic permutation table countermeasure.

Table 1. Timings obtained using a C implementation of AES without countermeasure, with masking countermeasure, and with proposed countermeasure: basic permutation table, and with time-memory trade-offs, on a 2 GHz laptop under Linux

Countermeasure	Timing (μs)	RAM (bytes)
AES encryption without countermeasure	2.2	214
AES encryption with masking	4.0	474
AES encryption with basic permutation tables	11	778
AES encryption with permutation tables + trade-offs	27	570
AES decryption without countermeasure	4.0	214
AES decryption with masking	9.5	474
AES decryption with basic permutation tables	23	842
AES decryption with permutation tables + trade-offs	42	570

8 Conclusion

We have described a new countermeasure against DPA for AES, that does not use random masking. Our countermeasure is provably secure against first order DPA, and reasonably efficient compared to the classical random masking countermeasure. We have also described three time-memory tradeoffs to reduce the RAM requirement; this can be useful for smart-card implementations.

Acknowledgments. Thanks to Christophe Clavier for suggesting the "Double and Add" approach in Section 6.3.

References

1. Akkar, M.L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162. Springer, Heidelberg (2001)
2. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)
3. Coron, J.S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
4. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
5. Golic, J.D., Tymen, C.: Multiplicative Masking and Power Analysis of AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523. Springer, Heidelberg (2003)
6. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666. Springer, Heidelberg (1999)
7. Kocher, P., et al.: DES and Other Cryptographic processes with Leak Minimization for smartcards and other cryptosystems. US 6,278,783 B1, June 3 (1999), `http://www.cryptography.com/technology/dpa/licensing.html`
8. IBM Corporation, Space-efficient, side-channel attack resistant table lookups. Application Patent 20030044003

9. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
10. Oswald, E., Schramm, K.: An Efficient Masking Scheme for AES Software Implementations. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 292–305. Springer, Heidelberg (2006)
11. Rao, J.R., Rohatgi, P., Scherzer, H., Tinguely, S.: Partitioning attacks: Or How to rapidly Clone Some GSM Cards. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy (2002)
12. Wolkerstorfer, J., Oswald, E., Lamberger, M.: An ASIC implementation of the AES SBoxes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271. Springer, Heidelberg (2002)

A AES Pseudo-code

```
Cipher(byte in[16], byte out[16], word w[44])
begin
   byte state[16]
   state=in
   AddRoundKey(state,w[0,3])
   for round=1 to 9
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state,w[round*4,(round+1)*4-1])
   end for
   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state,w[40,43])
end
```

Fig. 1. Pseudo-code for AES encryption

B List of Tables

We summarise in Table 2 the list of randomised tables required for each operation. Note that for decryption, we also need table S' to compute the key-schedule,

Table 2. Randomised tables required for each AES operation, for the basic permutation table countermeasure, and with the three time-memory tradeoffs

Operation	Tables required
AES encryption, basic	P, P^{-1}, XT_4^1, XT_4^2, S', ML, MH
AES decryption, basic	P, P^{-1}, XT_4^1, XT_4^2, S'^{-1}, ML, MH,RL, RH,TL, TH
AES encryption, tradeoffs	P, P^{-1}, XT_4^1, ML, MH, p_{12}, p_{21}, P', P'^{-1}, T
AES decryption, tradeoffs	P, P^{-1}, XT_4^1, ML, MH, p_{12}, p_{21}, P', P'^{-1}, T

but this table can be discarded at the end of the key-schedule. The RAM requirement for those randomised tables is summarised in Table 3.

```
InvCipher(byte in[16], byte out[16], word w[44])
begin
   byte state[16]
   state=in
   AddRoundKey(state,w[40,43])
   for round=9 downto 1
      InvShiftRows(state)
      InvSubBytes(state)
      AddRoundKey(state,w[round*4,(round+1)*4-1])
      InvMixColumns(state)
   end for
   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state,w[0,3])
end
```

Fig. 2. Pseudo-code for AES decryption

Table 3. Memory requirement for the randomised tables

Operation	RAM (bytes)
Permutations P and P^{-1}	32
Xor-table XT_4^1	128
Xor-table XT_4^2	128
Randomised SBOX S'	256
Randomised SBOX S'^{-1}	256
Tables ML, MH	32
Tables TL, TH, RL, RH	64
Permutations p_{12} and p_{21}	16
Permutations P' and P'^{-1}	32
Table T	128

Simplified Submission of Inputs to Protocols

Douglas Wikström*

CSC KTH Stockholm, Sweden
dog@csc.kth.se

Abstract. Consider an electronic election scheme implemented using a mix-net; a large number of voters submit their votes and then a smaller number of servers compute the result. The mix-net accepts an encrypted vote from each voter and outputs the set of votes in sorted order without revealing the permutation used. To ensure a fair election, the votes of corrupt voters should be independent of the votes of honest voters, i.e., some type of non-malleability or plaintext awareness is needed. However, for efficiency reasons the servers typically expect inputs from some homomorphic cryptosystem, which is inherently malleable.

In this paper we consider the problem of how non-malleability can be guaranteed in the submission phase and still allow the servers to start their computation with ciphertexts of the homomorphic cryptosystem. This can clearly be achieved using general techniques, but we would like a solution which is: (i) provably secure under standard assumptions, (ii) non-interactive for submittors (iii) very efficient for all parties in terms of computation and communication.

We give the first solution to this problem which has all these properties. Our solution is surprisingly simple and can be based on various Cramer-Shoup cryptosystems. To capture its security properties we introduce a variation of CCA2-security.

1 Introduction

Mix-Nets. A mix-net is a cryptographic protocol executed by N senders and k mix-servers, where typically N is quite large and k is fairly small, e.g., $N = 10^4$ and $k = 10$. The functionality implemented by a mix-net corresponds to a trusted party that collects inputs from the senders and then outputs the inputs in sorted order. The main application of mix-nets is for electronic elections. All known efficient robust mix-nets exploit the homomorphic properties of cryptosystems such as the El Gamal cryptosystem [16] in an essential way. A problem with using a homomorphic cryptosystem in the submission phase is that corrupted senders can submit inputs that are related to those of honest senders, i.e., the ciphertexts should be non-malleable during the submission phase and then become homomorphic when the mixing starts.

Formally, the proof of security fails if a semantically secure cryptosystem is used directly. When using the simulation paradigm, e.g., universally composable security [4], a mix-net is said to be secure if for every adversary attacking the mix-net

* Work partly done while at ETH Zürich, Department of Computer Science.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 293–308, 2008.

there is an ideal adversary (simulator), typically running the adversary as a black-box, attacking the ideal mix-net (the trusted party mentioned above) such that no environment can tell the two models apart. The simulator does not know the inputs of the honest parties and replaces them in its simulation, e.g., by zero messages. It must also extract the inputs of corrupted parties in its simulation and hand them to the ideal mix-net, since otherwise these inputs would be missing in the output from the ideal mix-net and the environment could trivially distinguish the two models. Any successful adversary must result in a successful attack against the underlying cryptosystem, which means that the simulator can not use the secret key of the cryptosystem to extract the inputs of corrupt senders.

General Case. More generally, consider a cryptographic protocol that starts with a submission phase where many parties submit ciphertexts formed with a public key pk, and a smaller group of servers hold shares of the secret key sk corresponding to pk. The servers then compute and publish some function of the input plaintexts. Typically, efficient protocols exploit the algebraic structure of the cryptosystem, e.g., homomorphic properties. The problem with using a semantically secure cryptosystem directly in such protocols is that it does not guarantee that the plaintexts submitted by corrupt parties are unrelated to those submitted by honest users.

Formally, the problem surfaces when the cryptographer constructing the protocol tries to reduce the security of his/her scheme to the security of the cryptosystem. If the simulation paradigm is used, some kind of simulator must be constructed and the simulator must extract the values of the corrupt parties to be able to hand these to an ideal version of the protocol. The existence of a successful adversary must contradict the security of the cryptosystem, i.e., extraction must be done without using the secret key of the cryptosystem. This is not possible using only a semantically secure cryptosystem.

Submission Problem. The submission problem is how to find a submission scheme such that: (I) the inputs of corrupted parties can be extracted by the simulator without using the secret key, and (II) its output is a list of ciphertexts of the form expected by the servers computing the result. These requirements are essential to allow use of the submission scheme as a prefix to the main protocol, but there are also several natural additional properties that we can look for, or even require, in the submission phase.

(i) The solution should be provably secure under standard assumptions in the plain model, i.e., without any random oracles or generic groups.
(ii) The submission of inputs should be non-interactive for submittors.
(iii) The solution should be very efficient for all parties in terms of computation and communication. More precisely, when N and k denotes the number of submittors and servers respectively, then the computational complexity of each submitter should be independent of k, the communication complexity of each server should be independent of N, and the computational complexity of each server should be of the form $T(k)+T'(N)$ for some functions T and T'.

1.1 Previous Work

Informally, we may view any solution to the problem as a form of proof of knowledge of the encrypted plaintext, since any solution must allow the simulator to extract the submitted plaintext without knowledge of the secret key of the semantically secure cryptosystem. We classify the solutions in the literature and some extensions as follows:

1. A non-interactive proof of knowledge in the *random oracle model* is used, either using the Naor and Yung double ciphertext trick, or with rewinding. Such solutions are typically very efficient, but unfortunately only heuristically secure [5]. Note that the CCA2-secure cryptosystems in the random oracle model given by Shoup and Gennaro [29] may be viewed as instantiations of this solution.
2. An *interactive* proof of knowledge [17] is used, either with a straight-line extractor in the public key setting using the Naor and Yung [20] double-ciphertext trick, or with a rewinding extractor.
3. A non-interactive proof of knowledge using *general techniques* [2] is used. This is *not efficient* for either the submittor or the servers, even using the recent techniques of Groth et al. [18]. One could also use the fairly general zero-knowledge proofs based on homomorphic encryption of Damgård, Fazio and Nicolosi [12], but this requires non-standard assumptions. Note that using a non-interactive proof in this way is essentially the construction of a CCA2-secure cryptosystem under general assumptions given by Sahai [27] based on the Naor-Yung double-ciphertext trick, but starting from a concrete semantically secure cryptosystem with useful structure.
4. A non-interactive (for the submitter) proof of knowledge based on verifiable secret sharing is used, for example using techniques from Abe, Cramer, and Fehr [1]. Then the *computational and communication complexity of the submitting party grows linearly* with the number of servers, since each server must receive an encrypted secret share, and the servers must interact for each submitted ciphertext to verify its proof.
5. Non-interactive secret-key proofs using Cramer and Damgård [7] could be used. Their technique allows a prover and verifier to set up a secret key in a preliminary phase that later allows the prover to show that it behaves in a way consistent with the secret keys. Their scheme could be used in two ways. Either each submittor would take part in a protocol in the preliminary phase where the needed correlated secret keys are generated, or the servers would generate secret keys relative each other that allow them to prove that they performed the verification of a Cramer-Shoup ciphertext correctly. In the former case, interaction is moved to a preliminary phase, but each submittor must still *interact* with the servers and the servers must store a secret key for each submittor. In the latter case, submitting is non-interactive for the submittor, but each server must send and receive a non-interactive proof for each sender, i.e., its *communication complexity* with the other servers is *linear* in N.

6. An arbitrary CCA2-secure cryptosystem is used and ciphertexts are translated into suitable semantically secure ciphertexts using *general multiparty computation* techniques. This is inefficient both in terms of computation and communication.
7. A special CCA2-secure cryptosystem such that a ciphertext can be transformed more easily into a new ciphertext for the basic semantically secure scheme needed by the servers is used. We list the solutions of this type we are aware of below.
   (a) Canetti and Goldwasser [6] and Lysyanskaya and Peikert [19] have given CCA2-secure cryptosystems with distributed decryption which allows transforming ciphertexts into ciphertexts for semantically secure cryptosystems. These either involve *interaction*, *expensive setup assumptions*, or only work for a *small number of servers*.
   (b) Boneh, Boyen, and Halevi [3] give a CCA2-secure cryptosystem with distributed decryption that may be viewed as containing a semantically secure cryptosystem, but its security is based on a *non-standard complexity assumption* based on pairings.
   (c) Cramer, Damgård, and Ishai [8] present a solution based on distributed pseudo random functions and share conversion that is reasonably efficient for a *small number of servers* and requires communication linear in the number of ciphertexts between the servers to verify validity.
8. Prabhakaran and Rosulek [25] present a re-randomizable and replayable CCA secure cryptosystem where one could view the transformation as trivial, i.e., nothing would be done with the ciphertexts before handing them to the underlying protocol.
   This work is interesting, but we are not aware of any (interesting) underlying protocol that accepts input ciphertexts of their cryptosystem. In fact, the authors point out that it can *not* be used directly to construct a mix-net, and even if that would be possible it would give an *inefficient* mix-net due to the larger and more complex ciphertexts.
   We remark that our work was publicly available [32] before their work was published.

To summarize, there are numerous solutions to the submission problem which satisfies properties (I) and (II), but no such solution has the properties (i)-(iii) listed above for any interesting underlying protocol.

What Is Used In Existing Mix-Nets? There are numerous proposed mix-nets with informal security arguments. If the submission problem is considered at all, the Fiat-Shamir heuristic is used (mostly even without a proof in the random oracle model). In the provably secure mix-nets either a secret sharing based solution is used [30], or an interactive proof of knowledge is used [31,33].

1.2 Our Contribution

We give a simple solution to the submission problem that is efficient both in terms of computation and communication. Although the solution is nothing

more than an observation on the Cramer-Shoup cryptosystem, it is novel and important, since it gives a truly *practical and provably secure* way for senders to submit their inputs to a mix-net, and this solution has eluded researchers ever since the Cramer-Shoup cryptosystem appeared 10 years ago.

The Idea. Recall the original Cramer and Shoup scheme [10]. The cryptosystem is deployed in a group G_q of prime order q in which the decision Diffie-Hellman assumption is assumed to be hard. The key generator first chooses two random generators $g_0, g_1 \in G_q$. Then it chooses $x_0, x_1, y_0, y_1, z \in \mathbb{Z}_q$ randomly and defines $c = g_0^{x_0} g_1^{x_1}$, $d = g_0^{y_0} g_1^{y_1}$, and $h = g_0^z$. It generates a collision-free hash function H. Finally, it outputs $(pk, sk) = ((H, g_0, g_1, c, d, h), (x_0, x_1, y_0, y_1, z))$. To encrypt a message $m \in G_q$ using the public key pk the encryption algorithm chooses $r \in \mathbb{Z}_q$ randomly and outputs $(u_0, u_1, e, v) = (g_0^r, g_1^r, h^r m, c^r d^{rH(u_0,u_1,e)})$. To decrypt a tuple $(u_0, u_1, e, v) \in G_q^4$ using the secret key sk the decryption algorithm tests if $u_0^{x_0} u_1^{x_1} (u_0^{y_0} u_1^{y_1})^{H(u_0,u_1,e)} = v$ to check the validity of the ciphertext. If so it outputs e/u_0^z, and otherwise the unit element of the group.[1]

Note that $h = g^z$ and z have the form of an El Gamal [16] public and secret key respectively and that (u_0, e) is nothing more than an El Gamal ciphertext. This is of course not a new observation. What seems to be a new observation is the fact that the holder of the secret key may reveal (x_0, x_1, y_0, y_1) without any loss in security as long as it never decrypts any ciphertext constructed after this point, and that this solves the submission problem.

Generalizing and Applying the Idea. To allow us to generalize the observation about the original Cramer-Shoup scheme and identify a class of cryptosystems for which it applies, we introduce the notion of an *augmented cryptosystem* which contains another cryptosystem as a component. In applications, the latter cryptosystem will have some useful structure, e.g., be homomorphic, that allows more efficient and simpler protocols. We also introduce a strengthened variation of CCA2-security called *submission security* and observe that the generic scheme of Cramer and Shoup [11] already satisfies this stronger definition. In the full version [32] we also illustrate the use of the new notion by applying it to general secure function evaluation, which strictly generalizes the notion of a mix-net.

The real efficiency gain from using our technique obviously depends on the application, but it is clear that when the number of submittors N is large the complexity of our solution based on the El Gamal cryptosystem is close to that of the most efficient heuristic solution in the random oracle model. Due to the cost of evaluating a pairing we also expect it to out-perform any solution based on elliptic curves with pairings.

Limitations of Our Approach. When using our solution, no new inputs can be accepted after part of the secret key is revealed. This is a minor drawback in the targeted applications, since we have a large number of submitting parties and executions of the underlying protocol are infrequent. When a new session

[1] In [10] a special symbol $\perp$ is output if the test fails, but this is only to simplify the analysis. Any fixed output works just as well.

is executed the servers simply generate a new key. However, it may be useful to re-use the public key of the basic cryptosystem in the underlying protocol. Thus, our definitions require that the augmentation can be replaced by a freshly generated augmentation without any loss in security. This allows using several independent augmentations that may be revealed sequentially, i.e., inputs can be processed in batches and then input to the same underlying protocol. We remark that for general threshold decryption, e.g. [6], our approach is not reasonable, since users requesting a decryption expect the result immediately.

1.3 Notation

We denote by PT and PPT the sets of deterministic and probabilistic polynomial time Turing machines respectively, and write PT^* for the set of non-uniform polynomial time Turing machines. We use n to denote the security parameter, and say that a function $\epsilon(n)$ is negligible if for every constant c there exists a constant n_0 such that $\epsilon(n) < n^{-c}$ for $n > n_0$. If pk is the public key of a cryptosystem, we denote by $\mathcal{M}_{pk}$, $\mathcal{C}_{pk}$, and $\mathcal{R}_{pk}$ the plaintext space, the ciphertext space, and the randomness space respectively.

2 Augmented Cryptosystems

Keeping our observation about the original Cramer-Shoup cryptosystem in mind, we formalize a general class of *augmented* cryptosystems that given part of the secret key allow conversion of a ciphertext into a ciphertext of another *basic* cryptosystem. In applications, the basic cryptosystem typically has special properties, e.g., it is homomorphic, that are exploited by the cryptographic protocol. We introduce the following definition.

Definition 1 (Augmented Cryptosystem). *A cryptosystem* $\mathcal{CS} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ *is an augmentation of a cryptosystem* $\mathcal{CS}^B = (\mathsf{Kg}^B, \mathsf{Enc}^B, \mathsf{Dec}^B)$ *if there exists an* augmentation algorithm $\mathsf{Aug} \in \mathrm{PPT}$ *and a* stripping algorithm $\mathsf{Strip} \in \mathrm{PT}$ *such that:*

1. *On input* 1^n, Kg *computes* $(pk^B, sk^B) = \mathsf{Kg}^B(1^n)$ *and* $(pk^A, sk^A) = \mathsf{Aug}(pk^B)$, *and outputs* $(pk, sk) = ((pk^A : pk^B), (sk^A : sk^B))$.
2. *On input* $((sk^A : sk^B), c)$, Dec *outputs* $\mathsf{Dec}^B_{sk^B}(\mathsf{Strip}_{pk,sk^A}(c))$.

Clearly, any cryptosystem can be viewed as a trivial augmentation of itself, and if it is CCA2-secure then the trivial augmentation is also submission secure as defined below, but we are interested in non-trivial augmentations where $\mathcal{CS}^B$ has structural properties useful in the construction of protocols.

Some readers may find it tempting to use a definition that mirrors the Cramer-Shoup cryptosystem more closely to avoid the existence of trivial augmentations, i.e., one could explicitly require that it is possible to check the "validity" of a ciphertext using sk^A. We remind those readers that for most cryptographic notions there are trivial instances, e.g., the identity map is a cryptosystem, and we see no reason to impose unnecessary conditions on which particular properties of the basic cryptosystem that should be considered useful.

2.1 Submission Security of Augmented Cryptosystems

Recall the game considered in the definition of CCA2-security [20,13,26]. The adversary is given a public key pk. Then it may ask any number of decryption queries to a decryption oracle $\mathsf{Dec}_{sk}(\cdot)$ holding the secret key sk corresponding to pk. The adversary must then choose two challenge messages m_0 and m_1. The game is parameterized by a bit b and returns a challenge ciphertext of the form $c = \mathsf{Enc}_{pk}(m_b)$. Then the adversary is again allowed to ask arbitrary decryption queries to $\mathsf{Dec}_{sk}(\cdot)$ with the exception of c, and must finally output a guess b' of the bit b. If the cryptosystem is CCA2-secure, then the difference in distribution of b' when $b = 0$ or $b = 1$ respectively, should be negligible. Consider the following game.

Experiment 1 (Submission Security, $\mathsf{Exp}^{\mathsf{sub}-b}_{\mathcal{CS},\mathcal{CS}^B,A}(n)$)

$$
\begin{aligned}
(pk^B, sk^B) &\leftarrow \mathsf{Kg}^B(1^n) && \textit{// Basic keys}\\
(pk^A_j, sk^A_j) &\leftarrow \mathsf{Aug}(pk^B) \quad \textit{for } j = 1, 2, 3, \ldots && \textit{// Augmentations}\\
(pk_j, sk_j) &\leftarrow \big((pk^A_j : pk^B), (sk^A_j : sk^B)\big) && \textit{// Augmented keys}\\
(i, m_0, m_1, state) &\leftarrow A^{pk^A_{(\cdot)}, sk^A_{(\cdot)}, \mathsf{Dec}_{sk_{(\cdot)}}(\cdot)}(\textit{choose}, pk^B) && \textit{// Choice of challenges}\\
c &\leftarrow \mathsf{Enc}_{pk_i}(m_b) && \textit{// Challenge ciphertext}\\
d &\leftarrow A^{pk^A_{(\cdot)}, sk^A_{(\cdot)}, \mathsf{Dec}_{sk_{(\cdot)}}(\cdot)}(\textit{guess}, state) && \textit{// Guess of adversary}
\end{aligned}
$$

The experiment returns 0 if $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$ was queried on (i, c) or if it was queried on (j, c') for some c' after $sk^A_{(\cdot)}$ was queried on j. Otherwise the experiment returns d.

In the game above, the adversary is given a public key pk^B of the basic cryptosystem. It can request that the experiment generates an augmentation $(pk^A_j, sk^A_j) = \mathsf{Aug}(pk^B)$, stores $(pk_j, sk_j) = ((pk^A_j : pk^B), (sk^A_j : sk^B))$, and returns $pk_j = (pk^A_j : pk^B)$ to the adversary. This is done by submitting the integer j to its $pk^A_{(\cdot)}$-oracle. Any subsequent identical queries j give the same pk_j. It can ask decryption queries. This is done by submitting an index and ciphertext pair (j, c') to its $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$-oracle. It can request that the experiment reveals an augmentation sk^A_j by submitting j to its $sk^A_{(\cdot)}$-oracle, but after such a query no more decryption queries of the form (j, c') for some ciphertext c' are allowed. Then the adversary must choose an index i and two challenge messages m_0 and m_1. The game is parameterized by a bit b and returns a challenge ciphertext of the form $c = \mathsf{Enc}_{pk_i}(m_b)$. The adversary is then again allowed to: ask for more fresh public keys, ask more decryption queries with the exception of decryption of (i, c), and request more augmentations or augmentation keys. Finally, it outputs a guess b' of b. If the cryptosystem is submission secure, then the difference in distributions of b' with $b = 0$ or $b = 1$ respectively should be negligible.

We could equivalently have defined a game where the game only generates an augmentation if requested to do so by the adversary, but the above is conceptually simpler.

Definition 2 (Submission Security). *An augmentation $\mathcal{CS}$ of $\mathcal{CS}^B$ is submission secure if* $\forall A \in \mathrm{PT}^*$*:* $|\Pr[\mathsf{Exp}^{\mathsf{sub}-0}_{\mathcal{CS},\mathcal{CS}^B,A}(n) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sub}-1}_{\mathcal{CS},\mathcal{CS}^B,A}(n) = 1]|$ *is negligible.*

Example 1. A simple example of a submission secure cryptosystem can be derived from the scheme of Sahai [27] based on the Naor and Yung double ciphertext trick [20]. A semantically secure cryptosystem $\mathcal{CS}^B$ is given and a CCA2-secure cryptosystem $\mathcal{CS} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is constructed as follows. To generate a public key, compute $(pk_0^B, sk_0^B) = \mathsf{Kg}^B(1^n)$ and $(pk_1^B, sk_1^B) = \mathsf{Kg}^B(1^n)$, and generate a common reference string CRS. Then output the key pair $(pk, sk) = ((pk_0^B : pk_1^B, CRS), (sk_0^B : sk_1^B))$. To encrypt a message m, output $(c_0, c_1, \pi) = (\mathsf{Enc}^B_{pk_0^B}(m), \mathsf{Enc}^B_{pk_1^B}(m), \pi)$, where π is a simulation sound non-interactive adaptively zero-knowledge proof (NIZKP) that the same message is encrypted in c_0 and c_1. To decrypt, verify the NIZKP and output $\mathsf{Dec}^B_{sk_0}(c_0)$ or 0 depending on if the NIZKP was accepted or not. The augmentation algorithm Aug takes (pk_1^B, CRS) as input and outputs $(pk_0^B, sk_0^B) = \mathsf{Kg}^B(1^n)$. The stripping algorithm Strip checks the NIZKP and outputs c_0 or $\mathsf{Enc}_{pk_0^B}(0, 0)$ depending on if the NIZKP was accepted or not.

3 Generic Cramer-Shoup Is Submission Secure

The fact that the generic CCA2-secure cryptosystem of Cramer and Shoup is submission secure if we view it as an augmentation of a basic semantically secure cryptosystem is quite easy to see from their security proof. On the other hand we need to *prove* that this is indeed the case. Thus, we recall their scheme and prove this fact, but we use coarse-grained and streamlined definitions. We also take the liberty of ignoring the technical problem of constructing efficiently computable hash families, since this complicates the definitions and does not add anything to our exposition (see [11] for details).

3.1 Preliminaries

Subset Membership Problems. A subset membership problem consists of three sets X, $L \subsetneq X$, and W, and a relation $R \subset X \times W$. The idea is that it should be hard to decide if an element is sampled from L or from $X \setminus L$. To be useful in cryptography we also need some algorithms that allow us to sample instances and elements, and check for membership in X.

Definition 3. *A subset membership problem $\mathbb{M}$ consists of a collection of distributions $(I_n)_{n\in\mathbb{N}}$, an instance generator* $\mathsf{Ins} \in \mathrm{PPT}$*, a sampling algorithm* $\mathsf{Sam} \in \mathrm{PPT}$*, and a membership checking algorithm* $\mathsf{Mem} \in \mathrm{PT}$ *such that:*

1. *I_n is a distribution on instance descriptions $\Lambda[X, L, W, R]$ specifying finite non-empty sets X, $L \subsetneq X$, and W, and a binary relation $R \subset X \times W$.*
2. *On input 1^n, Ins outputs an instance Λ with distribution statistically close to I_n.*
3. *On input 1^n and $\Lambda[X, L, W, R] \in [I_n]$ (the support of I_n), Sam outputs $(x, w) \in R$, where the distribution of x is statistically close to uniform over L.*
4. *On input 1^n, $\Lambda[X, L, W, R] \in [I_n]$, and $\zeta \in \{0,1\}^*$, Mem outputs 1 or 0 depending on if $\zeta \in X$ or not.*

Definition 4. *Let $\mathbb{M}$ be a subset membership problem. Sample Λ from I_n and let x and x' be randomly distributed over L and $X \setminus L$ respectively. We say that $\mathbb{M}$ is hard if for every $A \in \mathrm{PT}^*$: $|\Pr[A(\Lambda, x) = 1] - \Pr[A(\Lambda, x') = 1]|$ is negligible.*

Hash Families. Hash families are well known in the cryptographic literature and there are many variations. We assume that all families are indexed by a security parameter n.

Definition 5. *A projective hash family $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$ consists of finite non-empty sets K, X, $L \subsetneq X$, Π, and S, a function $\alpha : K \to S$, and a collection of hash functions $H = (H_k : X \times \Pi \to \Pi)_{k \in K}$, where $\alpha(k)$ determines H_k on $L \times \Pi$.*

Definition 6. *Let $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$ be a projective hash family, and let $k \in K$ be random. Then $\mathbb{H}$ is universal$_2$ if for every $s \in S$, $x, x' \in X$ with $x \notin L \cup \{x'\}$, and $\pi_x, \pi'_x, \pi, \pi' \in \Pi$, $\Pr_k[H_k(x, \pi_x) = \pi \wedge H_k(x', \pi'_x) = \pi' \mid \alpha(k) = s]$ is negligible.*

The following definition and lemma are stated informally in [11].

Experiment 2 (Computationally Universal$_2$, $\mathsf{Exp}^{\mathsf{cuni}_2}_{\mathbb{H},A}(n)$). *Let τ_k be the predicate defined by $\tau_k((x, \pi_x), \pi) \iff H_k(x, \pi_x) = \pi$, and consider the following experiment.*

$$\begin{aligned} k &\leftarrow K \\ (x, \pi_x, state) &\leftarrow A^{\tau_k(\cdot,\cdot)}(\alpha(k)) \\ &\leftarrow A^{\tau_k(\cdot,\cdot)}(H_k(x, \pi_x), state) \end{aligned}$$

Denote by $((x_i, \pi_{x,i}), \pi_i)$ the ith query to τ_k, and let i_l be the index of the last query before the first output. If A asks a query $((x_i, \pi_{x,i}), \pi_i)$ to τ_k with $H_k(x_i, \pi_{x,i}) = \pi_i$, $x_i \in X \setminus L$, and $i \leq i_l$ or $x_i \neq x$, then output 1 and otherwise 0.

Definition 7. *A projective hash family $\mathbb{H}$ is computationally universal$_2$ if for every $A \in \mathrm{PT}^*$, $\Pr[\mathsf{Exp}^{\mathsf{cuni}_2}_{\mathbb{H},A}(n) = 1]$ is negligible.*

Lemma 1. *If a projective hash family $\mathbb{H}$ is universal$_2$, then it is computationally universal$_2$.*

Proof. Denote by $((x_i, \pi_{x,i}), \pi_i)$ the ith query of A and let E_i be the event that $H_k(x_i, \pi_{x,i}) = \pi_i$, $x_i \in X \setminus L$, and $i \leq i_l$ or $x_i \neq x$. Condition on arbitrary fixed values of (x, π_x), $\pi = H_k(x, \pi_x)$, and $\alpha(k)$. Then the conditional probability of the event E_i is negligible by universiality$_2$ of $\mathbb{H}$. Since the fixed values are arbitrary, this holds also without conditioning. Finally, A asks at most a polynomial number of queries and the lemma follows from the union bound.

Definition 8. *Let $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$ be a projective hash family, and let $k \in K$, $x \in X \setminus L$, and $\pi \in X$ be random. Then $\mathbb{H}$ is smooth if for every $\pi_x \in \Pi$ the distributions of $(x, \pi_x, \alpha(k), H_k(x, \pi_x))$ and $(x, \pi_x, \alpha(k), \pi)$ are statistically close.*

Universal Hash Proof Systems. Informally, a hash proof system may be viewed as a non-interactive zero-knowledge proof system where only the holder of a secret key corresponding to the public common random string can verify a proof. Strictly speaking, the definition below corresponds, in the unconditional case, to a special case of what Cramer and Shoup [11] call "extended strongly (smooth, universal$_2$)" hash proof system.

Definition 9. *A (smooth, (computational) universal$_2$) hash proof system $\mathbb{P}$ for a subset membership problem $\mathbb{M}$ associates with each instance $\Lambda[X, L, W, R]$ a (smooth, (computationally) universal$_2$) projective hash family $\mathbb{H} = (H, K, X, L, \Pi, S, \alpha)$, and the following algorithms*

1. *A key generation algorithm $\mathsf{Gen} \in \mathrm{PPT}$ that on input 1^n and $\Lambda \in [I_n]$ (the support of I_n) outputs (s, k), where k is randomly distributed in K and $s = \alpha(k)$.*
2. *A private evaluation algorithm $\mathsf{PEval} \in \mathrm{PT}$ that on input 1^n, $\Lambda \in [I_n]$, $k \in K$, and $(x, \pi_x) \in X \times \Pi$ outputs $H_k(x, \pi_x)$.*
3. *A public evaluation algorithm $\mathsf{Eval} \in \mathrm{PT}$ that on input 1^n, $\Lambda \in [I_n]$, $\alpha(k)$ with $k \in K$, (x, π_x) and w, with $(x, w) \in R$ and $\pi_x \in \Pi$, outputs $H_k(x, \pi_x)$.*
4. *A membership checking algorithm $\mathsf{Mem} \in \mathrm{PT}$ that on input 1^n, $\Lambda \in [I_n]$, and $\zeta \in \{0,1\}^*$ outputs 1 or 0 depending on if $\zeta \in \Pi$ or not.*

3.2 Generic Scheme of Cramer and Shoup

Given the definitions above it is not hard to describe the generic cryptosystem of Cramer and Shoup [11]. Let $\mathbb{M}$ be a hard subset membership problem, such that Π can be fitted with a group operation for any Λ, let $\mathbb{P}_0 = (\mathsf{Gen}_0, \mathsf{PEval}_0, \mathsf{Eval}_0, \mathsf{Mem}_0)$ be a smooth hash proof system for $\mathbb{M}$, and let $\mathbb{P}_1 = (\mathsf{Gen}_1, \mathsf{PEval}_1, \mathsf{Eval}_1, \mathsf{Mem}_1)$ be a computationally universal$_2$ hash proof system for $\mathbb{M}$.

Key Generation. Compute $\Lambda[X, L, W, R] = \mathsf{Ins}(1^n)$, $(s, k) = \mathsf{Gen}_0(1^n, \Lambda)$, $(\widehat{s}, \widehat{k}) = \mathsf{Gen}_1(1^n, \Lambda)$, and output the key pair $(pk, sk) = ((\widehat{s} : \Lambda, s), (\widehat{k} : k))$.

Encryption of a message $m \in \Pi$**.** Compute $(x, w) = \mathsf{Sam}(\Lambda)$, $\pi = \mathsf{Eval}_0(\Lambda, s, x, w) = H_k(x)$, $e = m + \pi$, and $\widehat{\pi} = \mathsf{Eval}_1(\Lambda, \widehat{s}, x, w, e) = \widehat{H}_{\widehat{k}}(x, e)$, and output $(x, e, \widehat{\pi})$.

Decryption of a ciphertext $(x, e, \widehat{\pi})$**.** Output $m = e - \mathsf{PEval}_0(\Lambda, k, x) = e - H_k(x)$, only if $\mathsf{PEval}_1(\Lambda, \widehat{k}, x, e) = \widehat{H}_{\widehat{k}}(x, e) = \widehat{\pi}$ and otherwise output 0.

We have not modified the cryptosystem, except that we have introduced a colon in the notation to distinguish the two parts of the public and secret keys as needed in the definition of submission security, and we have replaced the special symbol $\perp$ by the zero plaintext.

Write $\mathcal{CS} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$, and let $\mathcal{CS}^B = (\mathsf{Kg}^B, \mathsf{Enc}^B, \mathsf{Dec}^B)$ be the underlying basic cryptosystem defined as follows. It has public key (Λ, s) and secret key k. A message $m \in \Pi$ is encrypted as (x, e), where $e = \mathsf{Eval}_0(\Lambda, s, x, w) + m$, and a ciphertext (x, e) is decrypted by computing $e - \mathsf{PEval}_0(\Lambda, k, x)$. It is clear that $\mathcal{CS}$ is an augmentation of $\mathcal{CS}^B$, since we can define $\mathsf{Aug}(\Lambda, s) = \mathsf{Gen}_1(1^n, \Lambda)$ and define $\mathsf{Strip}_{pk,\widehat{k}}(x, e, \widehat{\pi})$ to output (x, e) if $\mathsf{PEval}_1(\Lambda, \widehat{k}, x, e) = \widehat{\pi}$ and otherwise $\mathsf{Enc}_{pk^B}(0, 0)$.

Cramer and Shoup prove (see Theorem 1 in [11]) that $\mathcal{CS}$ is CCA2-secure under the assumption that $\mathbb{M}$ is hard. We prove a stronger result.

Proposition 1. *If $\mathbb{M}$ is a hard subset membership problem, then $\mathcal{CS}$ is a submission secure augmentation of $\mathcal{CS}^B$.*

The key observations needed to extend Cramer's and Shoup's proof of CCA2-security are:

1. The projective property of hash proofs implies that proofs computed using a witness and hash proofs computed using the private key $\widehat{k}$ are *identical* (indeed this is how a hash proof is verified). This means that the holder of $\widehat{k}$ can "perfectly simulate" proofs without the witness, i.e., *even if $\widehat{k}$ is made public* the "simulated proof" looks *exactly* like a proof computed using a witness.
2. The soundness of proofs computed by an adversary *before* $\widehat{k}$ is made public, is not decreased by the publishing of $\widehat{k}$.

The generic Cramer-Shoup scheme generalizes several concrete schemes described in [11], such as the El Gamal based scheme in the introduction, but also schemes based on the Paillier cryposystem [22]. Both schemes are common in efficient protocols.

3.3 Proof of Proposition 1

Conceptually, we follow the proof of Cramer and Shoup, but our proof is somewhat simplified, since we ignore the problem of approximating the hash families by efficiently computable hash families.

Denote by $T_b^{(0)}$ the machine that simulates the experiment $\mathsf{Exp}^{\mathsf{sub}-b}_{\mathcal{CS},\mathcal{CS}^B,A}(n)$ with some adversary $A \in \mathrm{PT}^*$, except that when computing the challenge ciphertext $(x, e, \widehat{\pi})$, the two hash proofs π and $\widehat{\pi}$ are computed as $\pi = \mathsf{PEval}_0(\Lambda, k, x) = H_k(x)$ and $\widehat{\pi} = \mathsf{PEval}_1(\Lambda, \widehat{k}_i, x, e) = \widehat{H}_{\widehat{k}_i}(x, e)$, where i is the challenge index chosen by the adversary. By the projectivity of hash proofs this does not change the distribution of the experiment.

We now change $T_b^{(0)}$ step by step until it is independent of b.

Claim 1. Denote by $T_b^{(1)}$ the machine $T_b^{(0)}$ except that x is chosen randomly in $X \setminus L$. Then $|\Pr[T_b^{(0)} = 1] - \Pr[T_b^{(1)} = 1]|$ is negligible.

Proof. Denote by A_{mem} an algorithm that tries to solve the subset membership problem $\mathbb{M}$. It accepts as input (Λ, x), where x either belongs to L or $X \setminus L$. It simulates $T_b^{(0)}$ except that it uses the instance Λ and defines the challenge ciphertext $(x, e, \widehat{\pi})$ using x from its input (Λ, x). Note that A_{mem} is identically distributed to $T_b^{(0)}$ or $T_b^{(1)}$ depending on if $x \in L$ or $x \in X \setminus L$. From the hardness of $\mathbb{M}$ follows that $|\Pr[T_b^{(0)} = 1] - \Pr[T_b^{(1)} = 1]|$ is negligible. □

Denote by $(i_j, (\pi_j, e_j, \widehat{\pi}_j))$ the jth query to the decryption oracle $\mathsf{Dec}_{sk_{(\cdot)}}(\cdot)$, and let j_l be the index of the last query before the adversary outputs its choice of challenge index and messages. Denote by $(x, e, \widehat{\pi})$ the challenge ciphertext, and let E be the event that A asks a decryption query $(i_j, (\pi_j, e_j, \widehat{\pi}_j))$ with $\widehat{H}_{\widehat{k}_{i_j}}(x_j, e_j) = \widehat{\pi}_j$, $x_j \in X \setminus L$, and $j \leq j_l$ or $x_j \neq x$, for some index j before it requests $sk^A_{i_j}$ from its $sk^A_{(\cdot)}$-oracle (it may of course not ever do this).

Claim 2. $\Pr[E]$ is negligible.

Proof. Let Q be the total number of queries to the augmentation oracle $pk_{(\cdot)}$ made by the adversary. Without loss we assume that the adversary asks the queries $l = 1, \ldots, Q$.

Define A^l_{uni} for $l = 1, \ldots, Q$ to be the machine that simulates $T_b^{(1)}$ and takes part in Experiment 2. The simulation is modified in that: $\widehat{s}_l$ is defined as the hash proof key received in Experiment 2, whenever $T_b^{(1)}$ needs to check a hash proof as $\mathsf{PEval}_1(\Lambda, \widehat{k}_l, x_j, e_j) = \widehat{H}_{\widehat{k}_l}(x_j, e_j)$ it simply queries its $\tau_{\widehat{k}_l}(\cdot, \cdot)$-oracle in Experiment 2 with (x_j, e_j) instead, and when it needs to compute $\mathsf{PEval}_1(\Lambda, \widehat{k}_l, x, e) = \widehat{H}_{\widehat{k}_l}(x, e) = \widehat{\pi}$ it outputs (x, e) and waits for $\widehat{H}_{\widehat{k}_l}(x, e) = \widehat{\pi}$ from the experiment instead. The computational universal$_2$ property and the union bound then implies the claim.

Note that the computational universal$_2$ property can be applied despite that the experiment reveals private hash proof keys, since by definition of submission security the adversary only wins if it never asks a decryption query after this point. This observation is the only essential change to the original proof. □

Denote by $T_b^{(2)}$ the machine $T_b^{(1)}$, except that it outputs $\perp$ if the event E occurs. The machine $T_b^{(2)}$ may not be efficient, but this does not matter since the remainder of the argument is statistical in nature.

Claim 3. Denote by $T_b^{(3)}$ the machine $T_b^{(2)}$ except that in the computation of the challenge ciphertext $(x, e, \widehat{\pi})$, π is chosen randomly in Π. Then $|\Pr[T_b^{(2)} = 1] - \Pr[T_b^{(3)} = 1]|$ is negligible.

Proof. Consider an arbitrary fixed instance Λ of the subset membership problem and an arbitrary fixed random string of the experiment conditioned on the event $\bar{E}$. Define a function $f : X \times S \times \Pi \to \{0, 1\}$ as follows. Let $f(x, \alpha(k), \pi)$ simulate $T_b^{(2)}$ except that the input parameters are used in the computation of the challenge ciphertext. Note that f exists, since $T_b^{(2)}$ outputs $\perp$ if the event E occurs and $\alpha(k)$ determines H_k on L by the projective property of $\mathbb{H}$, so the answers of all queries are determined by $\alpha(k)$. When $k \in K$, $x \in X$, and $\pi \in \Pi$ are randomly chosen, $f(x, \alpha(k), H_k(x))$ is identically distributed to $T_b^{(2)}$ and $f(x, \alpha(k), \pi)$ is identically distributed to $T_b^{(3)}$. The claim now follows from the smoothness of $\mathbb{P}$.

Conclusion of Proof of the Proposition. To conclude the proof of the proposition we simply note that the distributions of $T_0^{(3)}$ and $T_1^{(3)}$ are identical since Π is a group. The claims above now imply that $|\Pr[T_0^{(0)} = 1] - \Pr[T_1^{(0)} = 1]|$ is negligible.

4 Applications of Submission Security

The original motivation for this paper was to come up with a practical non-interactive submission phase in El Gamal based mix-nets. For readers that are not familiar with mix-nets we give an informal description of a construction that goes back to Sako and Kilian [28]. In the full version [32] we also illustrate how the notion of submission secure augmented cryptosystems can be used to construct and analyze the submission phase of a protocol in a modularized way for general secure function evaluation, and explain how this generalizes the mix-net setting in the informal description.

4.1 Informal Description of Application to a Mix-Net

There are many senders $S_1, \ldots, S_N$ and few mix-servers $M_1, \ldots, M_k$, e.g., $N = 10^4$ and $k = 10$. In a joint key generation phase the mix-servers generate a joint public key (g, h) such that each mix-server holds a verifiable secret share s_j of the joint secret key z such that $h = g^z$. This can be done using Feldman verifiable secret sharing [14]. To submit a message $m_i \in G_q$, a sender S_i computes an El Gamal ciphertext $(u_{0,i}, e_{0,i}) = (g^{r_i}, h^{r_i} m_i)$, where $r_i \in \mathbb{Z}_q$ is randomly chosen. Then the mix-servers take turns at re-encrypting, using the homomorphic property of El Gamal, and permuting the list of ciphertexts. In other words, for $j = 1, \ldots, k$, M_j computes and publishes $\{(u_{j,i}, e_{j,i})\} = \{(u_{j-1,\pi_j(i)} g^{t_{j,i}}, e_{j-1,\pi_j(i)} h^{t_{j,i}})\}$, where $t_{j,i} \in \mathbb{Z}_q$ and π_j are random. Finally, the mix-servers jointly and verifiably decrypt the list $\{(u_{k,i}, e_{k,i})\}$ output by the last mix-server M_k using their shares s_j, sort the result, and output it.

The idea is that due to the transformations computed by the mix-servers the correspondence between the output plaintexts and the input ciphertexts should be hidden. To ensure correctness, each mix-server also proves in zero-knowledge that it processed the list of ciphertexts correctly. This is done using a so called proof of a shuffle [21,15].

Unfortunately, the construction is completely insecure [24], since a malicious sender S_l may compute its ciphertext as $(u_{0,l}, e_{0,l}) = (u_{0,i}^a, e_{0,i}^a)$ for some random exponent a and then identify a matching pair (m_i, m_i^a) in the final output, where m_i is the message sent by S_i. This reveals the message m_i sent by the honest sender S_i. Intuitively, what is needed is a non-malleable cryptosystem, but on the other hand the cryptosystem must be homomorphic for re-encryption to be possible. Formally, what is needed in the overall proof of security of the mix-net (see [30,31,33]) is a way to extract the messages submitted by corrupted players without using the secret key of the cryptosystem, as explained in the introduction. In previous work this is either solved heuristically, or as in the cited works a proof of knowledge is used explicitly.

We augment the above to make the cryptosystem used for submission identical to the Cramer-Shoup scheme. We set $g_0 = g$ and let the mix-servers generate $g_1 \in G_q$, $x_0, x_1, y_0, y_1 \in \mathbb{Z}_q$, $c = g_0^{x_0} g_1^{x_1}$, and $d = g_0^{y_0} g_1^{y_1}$, where x_0, x_1, y_0, y_1 are verifiably secret shared among the mix-servers. This can be done using Pedersen verifiable secret sharing [23] and takes place after the joint key h is generated. This gives a Cramer-Shoup key pair $((H, g_1, c, d : g_0, h), (x_0, x_1, y_0, y_1 : z))$ with verifiably secret shared secret key. Due to the submission security of the cryptosystem the mix-servers may simply reconstruct the first part (x_0, x_1, y_0, y_1) of the shared key before starting the mixing process. This allows each mix-server to identify the valid ciphertexts *without any additional communication*, and form the list of El Gamal ciphertexts consisting of the El Gamal part of each valid ciphertext. Then the mix-servers process the El Gamal ciphertexts as explained above.

5 Future Work

In the mix-net application, all messages are free-form. This may not be the case in other applications. It is for example not the case in multi-candidate homomorphic election schemes, e.g., [9], where the submitted messages must be of a special form to encode a valid candidate. It is an interesting question if it is possible to come up with an efficient hash proof system that constrains the set of messages in this way. This would give a very efficient non-interactive submission phase for such election schemes in the standard model.

Acknowledgments

I thank Eike Kiltz for helpful discussions, and I thank Ronald Cramer for answering my questions about the relation between the generic Cramer-Shoup scheme and its concrete instantiations.

References

1. Abe, M., Cramer, R., Fehr, S.: Non-interactive distributed-verifier proofs and proving relations among commitments. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 206–223. Springer, Heidelberg (2002)
2. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: 20th ACM Symposium on the Theory of Computing (STOC), pp. 103–118. ACM Press, New York (1988)
3. Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001), `http://eprint.iacr.org`
5. Canetti, R., Goldreich, O., Halevi, S.: The random oracle model revisited. In: 30th ACM Symposium on the Theory of Computing (STOC), pp. 209–218. ACM Press, New York (1998)
6. Canetti, R., Goldwasser, S.: An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90–106. Springer, Heidelberg (1999)
7. Cramer, R., Damgård, I.: Secret-key zero-knowlegde and non-interactive verifiable exponentiation. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 223–237. Springer, Heidelberg (2004)
8. Cramer, R., Damgård, I., Ishai, Y.: Share conversion, pseudorandom secret-sharing and applications to secure computation. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 342–362. Springer, Heidelberg (2005)
9. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
10. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
11. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption (June 1999), `http://homepages.cwi.nl/~cramer/`
12. Damgård, I., Fazio, N., Nicolosi, A.: Non-interactive zero-knowledge from homomorphic encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 41–59. Springer, Heidelberg (2006)
13. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography. In: 23rd ACM Symposium on the Theory of Computing (STOC), pp. 542–552. ACM Press, New York (1991)
14. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 427–438. IEEE Computer Society Press, Los Alamitos (1987)
15. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
16. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)

17. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
18. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006)
19. Lysyanskaya, A., Peikert, C.: Adaptive security in the threshold setting: From cryptosystems to signature schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 331–350. Springer, Heidelberg (2001)
20. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM Symposium on the Theory of Computing (STOC), pp. 427–437. ACM Press, New York (1990)
21. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: 8th ACM Conference on Computer and Communications Security (CCS), pp. 116–125. ACM Press, New York (2001)
22. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
24. Pfitzmann, B., Pfitzmann, A.: How to break the direct RSA-implementation of mixes. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 373–381. Springer, Heidelberg (1990)
25. Prabhakaran, M., Rosulek, M.: Rerandomizable rcca encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 517–534. Springer, Heidelberg (2007)
26. Rackoff, C., Simon, D.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (1992)
27. Sahai, A.: Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In: 40th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 543–553. IEEE Computer Society Press, Los Alamitos (1999)
28. Sako, K., Killian, J.: Reciept-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
29. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998)
30. Wikström, D.: A universally composable mix-net. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 315–335. Springer, Heidelberg (2004)
31. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)
32. Wikström, D.: Simplified submission of inputs to protocols. Cryptology ePrint Archive, Report 2006/259 (2006), http://eprint.iacr.org/
33. Wikström, D., Groth, J.: An adaptively secure mix-net without erasures. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 276–287. Springer, Heidelberg (2006)

Unconditionally Reliable and Secure Message Transmission in Directed Networks Revisited

Arpita Patra, Ashish Choudhary*, and C. Pandu Rangan**

Dept of Computer Science and Engineering
IIT Madras, Chennai India 600036
arpita@cse.iitm.ernet.in, ashishc@cse.iitm.ernet.in, rangan@iitm.ernet.in

Abstract. In this paper, we design URMT and USMT protocols which are communication optimal in amortized sense and first of their kind.

Keywords: Error Probability, Information Theoretic Security.

1 Introduction

Consider the following problem: a sender $\mathbf{S}$ and a receiver $\mathbf{R}$ are a part of directed synchronous network and are connected by uni-directional vertex disjoint paths/channels (also called as *wires*), which are directed either from $\mathbf{S}$ to $\mathbf{R}$ or vice-versa. An adversary $\mathcal{A}_t$ having *unbounded computing power* controls at most t wires in Byzantine fashion. $\mathbf{S}$ intends to communicate a message $M^{\mathbf{S}}$ containing $\ell \geq 1$ field elements from a finite field $\mathbb{F}$ to $\mathbf{R}$. The challenge is to design a protocol such that after interacting in phases[1] as per the protocol, $\mathbf{R}$ should output $M^{\mathbf{R}}$ where $M^{\mathbf{R}} = M^{\mathbf{S}}$ with probability at least $1 - \text{poly}(\kappa)2^{-\kappa}$ and κ is the error parameter. This problem is called *unconditionally reliable message transmission* (URMT)[6,4]. The problem of *unconditionally secure message transmission* (USMT)[6,4] has an additional restriction that $\mathcal{A}_t$ should get *no* information about $M^{\mathbf{S}}$ in *information theoretic* sense. If $\mathbf{S}$ and $\mathbf{R}$ are directly connected by a private channel, as assumed in generic secure multiparty computation protocols [2,13,3,9], then reliable and secure communication between them is guaranteed. However when $\mathbf{S}$ and $\mathbf{R}$ are not adjacent then URMT/USMT protocols help to simulate a virtual reliable/secure link with very high probability.

Existing Literature: The problem of URMT and USMT in directed networks was first studied by Desmedt et al. [4]. Modeling the underlying network as a directed graph is well motivated because in practice not every communication channel admits bi-directional communication. For instance, a base-station may communicate to even a far-off hand-held device but the other way round communication may not be possible. Following the approach of Dolev et al.[5], the authors in [4] have abstracted the underlying directed network in the form of directed vertex disjoint paths/wires, which are directed either from $\mathbf{S}$ to $\mathbf{R}$ or

* Financial Support from Infosys Technology India Acknowledged.
** Work Supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation Sponsored by Department of Information Technology, Government of India.

[1] A phase is a send from $\mathbf{S}$ to $\mathbf{R}$ or vice-versa.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 309–326, 2008.

vice-versa. Under such settings, Desmedt et al. have shown that URMT/USMT tolerating $\mathcal{A}_t$ is possible iff there are total $2t+1$ wires between $\mathbf{S}$ and $\mathbf{R}$, of which at least $t+1$ should be directed from $\mathbf{S}$ to $\mathbf{R}$ [4]. Recently, Shankar et al. [10] have studied URMT in *arbitrary directed networks*, where they have given the complete characterization of URMT tolerating $\mathcal{A}_t$ by considering the underling directed network *as a whole*. Their characterization shows that it is inappropriate to model an underlying directed network in the form of directed wires between $\mathbf{S}$ and $\mathbf{R}$. However, it is likely to take exponential time to verify whether a given directed network and $\mathcal{A}_t$ satisfies the conditions given in [10] for the possibility of URMT. Moreover, as a part of their sufficiency condition, the authors in [10] have given an exponential time URMT protocol. These two shortcomings motivate us to relook at the *wire based* of Desmedt et al..

Network Model and Definitions: We follow the model of Desmedt et al. [4] and abstract the network in the form of a directed graph $G = (V, E)$, where $\mathbf{S}$ and $\mathbf{R}$ are two special honest nodes in V. We assume that there are n directed wires $f_1, f_2, \ldots, f_n$ from $\mathbf{S}$ to $\mathbf{R}$, called as *top band* and u directed wires $b_1, b_2, \ldots, b_u$ from $\mathbf{R}$ to $\mathbf{S}$, called as *bottom band*. Moreover, we assume that the wires in the *top band* are disjoint from the wires in the *bottom band*. Our results can be easily generalized for the case when these wires are not disjoint. A centralized adversary $\mathcal{A}_t$ with unbounded computing power actively controls at most t wires between $\mathbf{S}$ and $\mathbf{R}$, including the *top* and *bottom band* in a colluded fashion. The adversary is *adaptive* and its choice of corrupting a wire depends upon the data seen so far. A wire once under the control of $\mathcal{A}_t$, will remain so for the rest of the protocol. A wire under the control of $\mathcal{A}_t$ may behave arbitrarily and the communication over such wires is fully known and dictated by $\mathcal{A}_t$. We say that a wire is corrupted, if the value(s) sent over the wire is changed arbitrarily by $\mathcal{A}_t$. A wire which is not under the control of $\mathcal{A}_t$ is called *honest*. We assume that $n = max(t+1, 2t-u+1)$ and $n+u = 2t+1$, which is the *minimum* number of wires needed for the existence of URMT/USMT tolerating $\mathcal{A}_t$ [4]. The network is synchronous and a protocol is executed in phases. Our protocols provide *unconditional security*; i.e., *information theoretic security* with an error probability of at most $\text{poly}(\kappa)2^{-\kappa}$ in reliability, where κ is the error parameter. For this, all our computations are performed over a finite field $\mathbb{F}$ with $|\mathbb{F}| = \text{GF}(poly(\kappa)2^{\kappa})$. So each element is represented by $O(\kappa)$ bits. $\mathbf{M^S}$ denotes the message, which is a sequence of $\ell \geq 1$ elements from $\mathbb{F}$, that $\mathbf{S}$ intends to send to $\mathbf{R}$.

Our Contributions: One of the key parameters of any URMT/USMT protocol is its communication complexity. Though the USMT protocol of [4] is efficient, it is not optimal in terms of communication complexity. In this paper, we prove the lower bound on the communication complexity of multiphase URMT/USMT protocols[2]. Moreover, we show that our bounds are tight by giving efficient, poly-

[2] Any single phase URMT/USMT protocol in directed network is no different from a single phase URMT(USMT) protocol in undirected networks. So the connectivity requirement and lower bound on the communication complexity of single phase URMT and USMT in undirected networks [8] holds for directed networks also.

nomial time communication optimal URMT/USMT protocols which are first of their kind. Specifically, we show that (a) There exists an $O(u)$ phase URMT protocol, which reliably sends $\ell\kappa$ bits by communicating $O(\ell\kappa)$ bits for sufficiently large ℓ. (b) If at least one wire in the *bottom band* is un-corrupted, then there exists an $O(u)$ phase USMT protocol which securely sends $\ell\kappa$ bits by communicating $O(\ell\kappa)$ bits for sufficiently large ℓ. Thus in (a) and (b), we can achieve reliability and security respectively, with constant factor overhead in communication complexity in amortized sense. It is easy to see that the protocols in (a) and (b) are communication optimal. (c) If full bottom band is corrupted by $\mathcal{A}_t$, then any multiphase USMT protocol needs to communicate $\Omega\left(\frac{n\ell}{u}\kappa\right)$ bits to securely sends $(\ell\kappa)$ bits. Moreover, we show that the bound is tight.

Tools Used: 1. Unconditionally Reliable Authentication [9,4]**:** It is used to send a message M over a wire such that if the wire is uncorrupted, then $\mathbf{R}$ correctly gets M and if the wire is corrupted, then $\mathbf{R}$ does not get M but is able to detect the corruption with very high probability. This is done as follows: Let a non-zero $(a, b) \in_R \mathbb{F}^2$ is *securely* established between $\mathbf{S}$ and $\mathbf{R}$ in advance. $\mathbf{S}$ computes $x = URauth(M; a, b) = aM + b$ and sends (M, x) to $\mathbf{R}$ over the wire. Let $\mathbf{R}$ receives (M', x') along the wire. $\mathbf{R}$ verifies $x' \stackrel{?}{=} URauth(M'; a, b)$. If the test fails then $\mathbf{R}$ concludes that $M' \neq M$, otherwise $M' = M$. The tuple (a, b) is called *authentication key*. The probability that $M' \neq M$, but still $\mathbf{R}$ fails to detect it is at most $\frac{1}{|\mathbb{F}|}$, which is negligible in our context.

2. Unconditionally Secure Authentication [4]**:** Its goal is similar to $URauth$, except that M should be secure. This is done as follows: Let $(a, b, c) \in_R \mathbb{F}^3 - \{(0, 0, 0)\}$, which is *securely* established between $\mathbf{S}$ and $\mathbf{R}$ in advance. $\mathbf{S}$ computes $(x, y) = USauth(M; a, b, c) = (M + a, b(M + a) + c)$ and sends (x, y) to $\mathbf{R}$ over the wire. Let $\mathbf{R}$ receives (x', y') along the wire. $\mathbf{R}$ verifies $y' \stackrel{?}{=} bx' + c$. If the test fails then $\mathbf{R}$ concludes that wire is corrupted, else $\mathbf{R}$ recovers $x' - a$. It is easy to see that M is information theoretic secure. Moreover, if $(x', y') \neq (x, y)$, then $\mathbf{R}$ will detect it with probability $\geq (1 - \frac{1}{|\mathbb{F}|})$.

3. Unconditional Hashing [1]**:** Let $(v_1, v_2, \ldots, v_\ell) \in \mathbb{F}^\ell$ and $k \in \mathbb{F} - \{0\}$. Then we define $hash(k; v_1, v_2, \ldots, v_\ell) = v_1 + v_2k + v_3k^2 + \ldots + v_\ell k^{\ell-1}$. Here k is called the *hash key*. The probability that two different vectors map to the same hash value for an uniformly chosen hash key is at most $\frac{\ell}{|\mathbb{F}|}$. If $\mathcal{A}_t$ knows only k and $hash(k; v_1, v_2, \ldots, v_\ell)$, then first $\ell - 1$ elements in the vector will be secure.

4. Extracting Randomness [11]**:** Suppose $\mathbf{S}$ and $\mathbf{R}$ by some means agree on $x = [x_1\ x_2\ \ldots\ x_n] \in \mathbb{F}^n$ such that $\mathcal{A}_t$ knows $n - f$ components of x, but has no information about the other f components of x. However $\mathbf{S}$ and $\mathbf{R}$ do not know which values are known to $\mathcal{A}_t$. The goal of $\mathbf{S}$ and $\mathbf{R}$ is to agree on $[y_1\ y_2\ \ldots\ y_f] \in \mathbb{F}^f$, such that $\mathcal{A}_t$ has no information about $[y_1\ y_2\ \ldots\ y_f]$. This is done as follows:

Algorithm EXTRAND$_{n,f}(x)$: Let V be a publicly known $n \times f$ Vandermonde matrix with members in $\mathbb{F}$. $\mathbf{S}$ and $\mathbf{R}$ locally computes $[y_1\ y_2\ \ldots\ y_f] = [x_1\ x_2\ \ldots\ x_n]V$.

2 Three Phase USMT Protocol of Desmedt et al. [4,12]

We briefly recall the three phase USMT protocol of [4] to send a message $m^{\mathbf{S}} \in \mathbb{F}$. We call the protocol as $\Pi^{Existing}$. We recall the protocol to highlight few techniques which are also used in our URMT and USMT protocols. In the protocol, there are two cases: (a) There exists $t+1$ non-faulty wires in the top band; (b) There exists less than $t+1$ non-faulty wires in the top band. During **Phase I**, **S** constructs $(t+1)$-out-of-n secret shares of $m^{\mathbf{S}}$ and associates one share with one wire in the top band. In order to authenticate the share associated with a wire, **S** selects n pair of random authentication keys. **S** then sends to **R** the share associated with a wire, authenticated with all the n keys. Parallely, **S** sends the authentication keys to **R**, one over each wire. In addition, **S** associates a random three tuple with each wire and sends it to **R**. If there are $t+1$ non-faulty wires in the top band, then at the end of **Phase I**, **R** will get at least $t+1$ correct shares with which he can recover $m^{\mathbf{R}}$.

If **R** cannot recover $m^{\mathbf{R}}$ at the end of **Phase I**, then it implies that there is at least one honest wire in the bottom band. So using the bottom band, **S** and **R** tries to correctly and securely agree on a shared authentication key and encryption key to securely communicate $m^{\mathbf{S}}$. For this, **R** uses the 3-tuples $(a_i^{\mathbf{R}}, b_i^{\mathbf{R}}, c_i^{\mathbf{R}})$ received from **S**. Now **R** sends a random non-zero 2-tuple $(d_i^{\mathbf{R}}, e_i^{\mathbf{R}})$ to **S** on each wire in bottom band. In addition, each such 2-tuple is authenticated by u random non-zero keys. Now according to the values received from **R**, **S** divides the bottom band into sub-sets $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$, where $k \leq u$, such that for each $1 \leq l \leq k$, all the wires in $\mathcal{B}_l$ behave in a "consistent" way. In particular, there exists at least one path set $\mathcal{B}_l$ that behave honestly during **Phase II**. Though **S** cannot determine which path set was honest, **S** will try to use each of them in a separate way and let **R** to determine which path set is honest. In **Phase II**, $\langle \ldots \rangle$ denotes a function used in [4], which maps a variable size (the variable size is bounded by a pre-defined bound) ordered subset of $\mathbb{F}$ to an image element in a field extension $\mathbb{F}^*$. The **Phase II** is as follows:

Table 1. Phase I of USMT Protocol $\Pi^{Existing}$[4]

1. **S** selects a random polynomial $p(x)$ of degree t over $\mathbb{F}$ such that $p(0) = m^{\mathbf{S}}$ and computes the secret shares $(s_1^{\mathbf{S}}, s_2^{\mathbf{S}}, \ldots, s_n^{\mathbf{S}})$, where $s_i^{\mathbf{S}} = p(i)$, $1 \leq i \leq n$. In order to authenticate each $s_i^{\mathbf{S}}$, **S** selects n random non-zero authentication keys $(a_{i,j}^{\mathbf{S}}, b_{i,j}^{\mathbf{S}}) \in \mathbb{F}^2$, $1 \leq j \leq n$. In addition, corresponding to each wire f_i in *top band*, **S** selects a random non-zero three tuple $(a_i^{\mathbf{S}}, b_i^{\mathbf{S}}, c_i^{\mathbf{S}}) \in \mathbb{F}^3$.
2. **S** sends $\{s_i^{\mathbf{S}}, d_{i,1}^{\mathbf{S}}, d_{i,2}^{\mathbf{S}}, \ldots, d_{i,n}^{\mathbf{S}}\}$ and the three tuple $(a_i^{\mathbf{S}}, b_i^{\mathbf{S}}, c_i^{\mathbf{S}})$ to **R** through wire f_i where $d_{i,j} = URauth(s_i^{\mathbf{S}}; a_{i,j}^{\mathbf{S}}, b_{i,j}^{\mathbf{S}}), 1 \leq j \leq n$. In addition, **S** sends the authentication key $(a_{i,j}^{\mathbf{S}}, b_{i,j}^{\mathbf{S}})$ to **R** through wire f_j, for $1 \leq j \leq n$.

Computation by R at the end of Phase I:

1. Let **R** receives $\{s_i^{\mathbf{R}}, d_{i,1}^{\mathbf{R}}, d_{i,2}^{\mathbf{R}}, \ldots, d_{i,n}^{\mathbf{R}}\}$ and $(a_i^{\mathbf{R}}, b_i^{\mathbf{R}}, c_i^{\mathbf{R}})$ along wire f_i and keys $(a_{i,j}^{\mathbf{R}}, b_{i,j}^{\mathbf{R}})$ along wire f_j. **R** computes $Support_i = |\{j : d_{i,j}^{\mathbf{R}} = URauth(s_i^{\mathbf{R}}; a_{i,j}^{\mathbf{R}}, b_{i,j}^{\mathbf{R}})\}|$. If $Support_i \geq t+1$, then **R** concludes that $s_i^{\mathbf{R}}$ is a valid share. Otherwise, it is an invalid share. If **R** receives $t+1$ valid shares then **R** recovers the secret $m^{\mathbf{R}}$ from these valid shares and terminates. Otherwise, **R** proceeds to execute **Phase II**.

1. For $1 \leq i \leq n$, $\mathbf{R}$ chooses a random non-zero $r_i^{\mathbf{R}} \in \mathbb{F}$ and computes $\beta^{\mathbf{R}} = \{(r_1^{\mathbf{R}}, \gamma_1^{\mathbf{R}}), (r_2^{\mathbf{R}}, \gamma_2^{\mathbf{R}}), \ldots, (r_n^{\mathbf{R}}, \gamma_n^{\mathbf{R}})\}$, where $\gamma_j^{\mathbf{R}} = hash(r_j^{\mathbf{R}}; a_j^{\mathbf{R}}, b_j^{\mathbf{R}}, c_j^{\mathbf{R}}), 1 \leq j \leq n$. For each $1 \leq i \leq u$, $\mathbf{R}$ selects a random non-zero 2-tuple $(d_i^{\mathbf{R}}, e_i^{\mathbf{R}}) \in \mathbb{F}^2$. In order to authenticate $(d_i^{\mathbf{R}}, e_i^{\mathbf{R}})$, $\mathbf{R}$ selects u random non-zero keys $\{(v_{i,j}^{\mathbf{R}}, w_{i,j}^{\mathbf{R}}) \in \mathbb{F}^2 : 1 \leq j \leq u\}$.

2. For each $1 \leq i \leq u$, $\mathbf{R}$ sends $\beta^{\mathbf{R}}, (d_i^{\mathbf{R}}, e_i^{\mathbf{R}})$ and $\{\alpha_{i,j}^{\mathbf{R}} : 1 \leq j \leq n\}$, where $\alpha_{i,j}^{\mathbf{R}} = URauth(\langle d_i^{\mathbf{R}}, e_i^{\mathbf{R}} \rangle; v_{i,j}^{\mathbf{R}}, w_{i,j}^{\mathbf{R}}) : 1 \leq j \leq u\}$ to $\mathbf{S}$ via wire b_i and the keys $(v_{i,j}^{\mathbf{R}}, w_{i,j}^{\mathbf{R}})$ to $\mathbf{S}$ via wire b_j for each $1 \leq j \leq u$.

Computation by S at the end of Phase II: 1. Let $\mathbf{S}$ receives $\beta_i^{\mathbf{S}}, (d_i^{\mathbf{S}}, e_i^{\mathbf{S}})$ and $\{\alpha_{i,j}^{\mathbf{S}} : 1 \leq j \leq u\}$ from $\mathbf{R}$ via wire b_i and $(v_{i,j}^{\mathbf{S}}, w_{i,j}^{\mathbf{S}})$ from $\mathbf{R}$ via wire b_j for each $1 \leq j \leq u$. $\mathbf{S}$ divides the bottom band $\{b_1, b_2, \ldots, b_u\}$ into subsets $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$, where $k \leq u$, such that for any l, m, p with $1 \leq l \leq k, 1 \leq m, p \leq u$ and $b_m, b_p \in \mathcal{B}_l$, we have: (a) $\beta_m^{\mathbf{S}} = \beta_p^{\mathbf{S}}$; (b) $\alpha_{m,p}^{\mathbf{S}} = URauth(\langle d_m^{\mathbf{S}}, e_m^{\mathbf{S}} \rangle; v_{m,p}^{\mathbf{S}}, w_{m,p}^{\mathbf{S}})$; (c) $\alpha_{p,m}^{\mathbf{S}} = URauth(\langle d_p^{\mathbf{S}}, e_p^{\mathbf{S}} \rangle; v_{p,m}^{\mathbf{S}}, w_{p,m}^{\mathbf{S}})$.

2. For $\mathcal{B}_l$, let $b_m \in \mathcal{B}_l$ and $\beta_m^{\mathbf{S}} = \{(r_{i,l}^{\mathbf{S}}, \gamma_{i,l}^{\mathbf{S}}) : 1 \leq i \leq n\}$. $\mathbf{S}$ computes the set

$$\mathcal{F}_l = \{i : \gamma_{i,l}^{\mathbf{S}} = hash(r_{i,l}^{\mathbf{S}}; a_i^{\mathbf{S}}, b_i^{\mathbf{S}}, c_i^{\mathbf{S}}), 1 \leq i \leq n\}$$

If $|\mathcal{B}_l| + |\mathcal{F}_l| \leq t$ then $\mathbf{S}$ decides that $\mathcal{B}_l$ is unacceptable set, otherwise $\mathcal{B}_l$ is acceptable.

From the properties of $URauth$ and $hash$, it is easy to check that the following holds: (a) If b_i is an honest wire in the bottom band and $b_i \in \mathcal{B}_l$, then with very high probability, the random 2-tuples that $\mathbf{S}$ has received along the wires in $\mathcal{B}_l$ are not modified; (b) If b_i is an honest wire in the bottom band and $b_i \in \mathcal{B}_l$, then $\mathcal{B}_l$ is an acceptable set. However, all acceptable sets may look same to $\mathbf{S}$ and $\mathbf{S}$ may not determine whether an acceptable set contains all honest wires. In the worst case, $\mathcal{A}_t$ can control the bottom band in such a way that there are u $\mathcal{B}_l$'s, with one wire from the bottom band in each $\mathcal{B}_l$.

Table 2. Phase III and Secret Recovery in Protocol $\Pi^{Existing}$

Phase III: S to R: For each acceptable set $\mathcal{B}_l$ and the corresponding set $\mathcal{F}_l$, $\mathbf{S}$ does the following:

- From the wires in $\mathcal{F}_l$ and $\mathcal{B}_l$, $\mathbf{S}$ computes his version of the keys $\mathcal{C}_l^{\mathbf{S}} = \sum_{f_i \in \mathcal{F}_l} a_i^{\mathbf{S}} + \sum_{b_i \in \mathcal{B}_l} d_i^{\mathbf{S}}$ and $\mathcal{D}_l^{\mathbf{S}} = \sum_{f_i \in \mathcal{F}_l} b_i^{\mathbf{S}} + \sum_{b_i \in \mathcal{B}_l} e_i^{\mathbf{S}}$. $\mathbf{S}$ then sends $(\psi_l^{\mathbf{S}}, \lambda_l^{\mathbf{S}})$ to $\mathbf{R}$ over all the wires in $\mathcal{F}_l$, where $\psi_l^{\mathbf{S}} = \langle \mathcal{B}_l, \mathcal{F}_l, m^{\mathbf{S}} + \mathcal{C}_l^{\mathbf{S}} \rangle$ and $\lambda_l^{\mathbf{S}} = URauth(\psi_l^{\mathbf{S}}; \mathcal{C}_l^{\mathbf{S}}, \mathcal{D}_l^{\mathbf{S}})$.

Message Recovery by R: $\mathbf{R}$ knows that in the worst case, $\mathbf{S}$ could have sent u 2-tuples over each wire in the top band, corresponding to the case when there are u acceptable sets. Let $\mathbf{R}$ receives $(\psi_{i,l}^{\mathbf{R}}, \lambda_{i,l}^{\mathbf{R}})$ over wire f_i for $1 \leq i \leq n$ and $1 \leq l \leq u$.

1. For each $1 \leq i \leq n$, $\mathbf{R}$ computes $\langle \mathcal{B}_{i,l}^{\mathbf{R}}, \mathcal{F}_{i,l}^{\mathbf{R}}, \tau_{i,l}^{\mathbf{R}} \rangle = \psi_{i,l}^{\mathbf{R}}$ (that is, $\mathbf{R}$ decomposes $\psi_{i,l}^{\mathbf{R}}$). $\mathbf{R}$ then computes his version of the keys $\mathcal{C}_{i,l}^{\mathbf{R}} = \sum_{f_j \in \mathcal{F}_{i,l}} a_j^{\mathbf{R}} + \sum_{b_j \in \mathcal{B}_{i,l}} d_j^{\mathbf{R}}$ and $\mathcal{D}_{i,l}^{\mathbf{R}} = \sum_{f_j \in \mathcal{F}_{i,l}} b_j^{\mathbf{R}} + \sum_{b_j \in \mathcal{B}_{i,l}} e_j^{\mathbf{R}}$.
2. For $1 \leq i \leq n$, $\mathbf{R}$ checks whether $\lambda_{i,l}^{\mathbf{R}} \stackrel{?}{=} URauth(\psi_{i,l}^{\mathbf{R}}; \mathcal{C}_{i,l}^{\mathbf{R}}, \mathcal{D}_{i,l}^{\mathbf{R}})$. If the equation holds then $\mathbf{R}$ computes the secret $m^{\mathbf{R}} = \tau_{i,l}^{\mathbf{R}} - \mathcal{C}_{i,l}^{\mathbf{R}}$ and terminates.

S continues the protocol by assuming that each acceptable set is correct. In other words, assuming that all the wires in an acceptable set $\mathcal{B}_l$ are non-faulty, **S** determines which of the random 3-tuples $(a_i^{\mathbf{S}}, b_i^{\mathbf{S}}, c_i^{\mathbf{S}})$, have been correctly received by **R**. Using these "correctly-received-by-**R**" 3-tuples and the random 2-tuples received by **S** via the wires in $\mathcal{B}_l$, **S** computes the authentication key and encryption key to securely send the messages to **R**. If the assumption that $\mathcal{B}_l$ contains only non-faulty wires is valid, then **R** would be able to compute the same authentication and encryption key. Since at least one of the acceptable path set is non-faulty, **R** will be able to decrypt the secret message correctly. The **Phase III** is shown in Table 2. It is easy to check that with very high probability, $m^{\mathbf{R}} = m^{\mathbf{S}}$. Since, for an acceptable set $\mathcal{B}_l$, $|\mathcal{F}_l| + |\mathcal{B}_l| > t$, the adversary learns no information about $\mathcal{C}_l^{\mathbf{S}}$ or $\mathcal{D}_l^{\mathbf{S}}$ and hence about $m^{\mathbf{S}}$.

Modified Version of Desmedt's USMT Protocol: We now present a modified version of protocol $\Pi^{Existing}$, called $\Pi^{Existing}_{modified}$, where all the computation and communication is done in $\mathbb{F}$. The purpose of presenting $\Pi^{Existing}_{modified}$ is to introduce certain new techniques, which we have also used in our later protocols. Protocol $\Pi^{Existing}_{modified}$ will be used as a sub-protocol in our communication optimal URMT and USMT protocols. The protocol securely sends a message $m^{\mathbf{S}} = \{m_1^{\mathbf{S}}\ m_2^{\mathbf{S}}\ \ldots\ m_{\frac{n}{3}}^{\mathbf{S}}\}$ containing $\frac{n}{3} = \Theta(n)$ elements from $\mathbb{F}$ by communicating $O(n^3)$ elements from $\mathbb{F}$. During **Phase I**, **S** selects a random polynomial $M^{\mathbf{S}}(x)$ over $\mathbb{F}$ of degree $n-1+t$ such that the lower order $\frac{n}{3}$ coefficients of $M^{\mathbf{S}}(x)$ are elements of $m^{\mathbf{S}}$. **S** then computes $M^{\mathbf{S}}(1), M^{\mathbf{S}}(2), \ldots, M^{\mathbf{S}}(n+t)$. **S** selects $n+t$ random polynomials $f_1^{\mathbf{S}}(x), f_2^{\mathbf{S}}(x), \ldots, f_{n+t}^{\mathbf{S}}(x)$ over $\mathbb{F}$, each of degree t, such that $f_i^{\mathbf{S}}(0) = M^{\mathbf{S}}(i), 1 \leq i \leq n+t$. **S** then evaluates each $f_i^{\mathbf{S}}(x)$ at $x = 1, 2, \ldots, n$ to form an n tuple $f_i^{\mathbf{S}} = [f_i^{\mathbf{S}}(1)\ f_i^{\mathbf{S}}(2)\ \ldots\ f_i^{\mathbf{S}}(n)]$. **S** now constructs an $(n) \times (n+t)$ matrix T where i^{th} column of T contains the n tuple $f_i^{\mathbf{S}}, 1 \leq i \leq n+t$. Let $F_j^{\mathbf{S}} = [f_1^{\mathbf{S}}(j)\ f_2^{\mathbf{S}}(j)\ \ldots\ f_{n+t}^{\mathbf{S}}(j)]$ denotes the $j^{th}, 1 \leq j \leq n$ row of T. Now **Phase I** is as follows:

Phase I: S to R: Along wire $f_j, 1 \leq j \leq n$, **S** sends the following to **R**: (a) The vector $F_j^{\mathbf{S}}$, a random non-zero *hash key* $\alpha_j^{\mathbf{S}}$ and the n tuple $[v_{1j}^{\mathbf{S}}\ v_{2j}^{\mathbf{S}}\ \ldots\ v_{nj}^{\mathbf{S}}]$, where $v_{ij}^{\mathbf{S}} = hash(\alpha_j^{\mathbf{S}}; F_i^{\mathbf{S}}), 1 \leq i \leq n$; (b) A random non-zero $(n+1)$ tuple $(x_{1,j}^{\mathbf{S}}, x_{2,j}^{\mathbf{S}}, \ldots, x_{n+1,j}^{\mathbf{S}})$, which is independent of $F_j^{\mathbf{S}}$.

Computation by R at the end of Phase I: Let **R** receives the vector $F_j^{\mathbf{R}}$, *hash key* $\alpha_j^{\mathbf{R}}$, the n tuple $[v_{1j}^{\mathbf{R}}\ v_{2j}^{\mathbf{R}}\ \ldots\ v_{nj}^{\mathbf{R}}]$ and the $n+1$ tuple $(x_{1,j}^{\mathbf{R}}, x_{2,j}^{\mathbf{R}}, \ldots, x_{n+1,j}^{\mathbf{R}})$ along wire $f_j, 1 \leq j \leq n$.

1. For $1 \leq j \leq n$, **R** computes $Support_j = |\{f_i : v_{ji}^{\mathbf{R}} = hash(\alpha_i^{\mathbf{R}}; F_j^{\mathbf{R}})\}|$. If $Support_j \geq t+1$, then **R** concludes that $F_j^{\mathbf{R}}$ is a valid row of T. Otherwise, **R** concludes that $F_j^{\mathbf{R}}$ is an invalid row of T.
2. If **R** has received $t+1$ valid rows, then **R** reconstructs the secret $m^{\mathbf{R}}$ from them and terminates protocol (see Theorem 1). Otherwise, **R** proceeds to execute **Phase II**.

Lemma 1. *If $F_j^{\mathbf{R}}$ is a valid row, then with overwhelming probability $F_j^{\mathbf{R}} = F_j^{\mathbf{S}}$.*

Lemma 2. *During* **Phase I**, *at least n coefficients of $M^{\mathbf{S}}(x)$ are information theoretically secure.*

Theorem 1. *If* $\mathbf{R}$ *gets* $t+1$ *valid rows then* $\mathbf{R}$ *can securely recover* $m^{\mathbf{S}}$ *with very high probability.*

For complete proof of Lemma 1, Lemma 2 and Theorem 1, please see the full version of the paper [7]. □

If $\mathbf{R}$ does not get $t+1$ valid rows, then $\mathbf{R}$ concludes that at least one wire in the bottom band is honest. So $\mathbf{R}$ proceeds to execute **Phase II** as shown in Table 3. **Phase II** is similar to the **Phase II** of protocol $\Pi^{Existing}$, except that $\beta^{\mathbf{R}}$ contains the hashed value of each $n+1$ tuple received from $\mathbf{S}$. Moreover, along each wire in the bottom band, $\mathbf{R}$ now sends an $(n+u)$ tuple and hash it with u random keys. Now as in protocol $\Pi^{Existing}$, depending upon the values received along the wires in the bottom band, $\mathbf{S}$ divides the bottom band into different subsets. As in the previous protocol, it is straightforward to check that the following holds: (a) If b_i is an honest wire in the bottom band and $b_i \in \mathcal{B}_l$, then with very high probability, the random $(n+u)$-tuples that $\mathbf{S}$ has received along the wires in $\mathcal{B}_l$ are not modified; (b) If b_i is an honest wire in the bottom band and $b_i \in \mathcal{B}_l$, then $\mathcal{B}_l$ is an acceptable set.

Table 3. Phase II and computation by $\mathbf{S}$ at the end of **Phase II** in $\Pi^{Existing}_{modified}$

Phase II: R to S (if R has not recovered the secret at the end of Phase I)

1. For each $1 \leq j \leq n$, $\mathbf{R}$ chooses a random non-zero hash key $r_j^{\mathbf{R}} \in \mathbb{F}$ and computes the set $\beta^{\mathbf{R}} = \{(r_j^{\mathbf{R}}, \gamma_j^{\mathbf{R}}) : 1 \leq j \leq n\}$, where $\gamma_j^{\mathbf{R}} = hash(r_j^{\mathbf{R}}; x_{1,j}^{\mathbf{R}}, x_{2,j}^{\mathbf{R}}, \ldots, x_{n+1,j}^{\mathbf{R}})$.
2. For each $1 \leq j \leq u$, $\mathbf{R}$ selects a random non-zero $n+u$ tuple $(y_{1,j}^{\mathbf{R}}, y_{2,j}^{\mathbf{R}}, \ldots, y_{n+u,j}^{\mathbf{R}}) \in \mathbb{F}^{n+u}$. In order to hash each such n tuple, $\mathbf{R}$ selects u random non-zero keys $\{key_{i,j}^{\mathbf{R}} : 1 \leq i \leq u\}$ from $\mathbb{F}$.
3. For each $1 \leq j \leq u$, $\mathbf{R}$ sends $\beta^{\mathbf{R}}$ and the $n+u$-tuple $(y_{1,j}^{\mathbf{R}}, y_{2,j}^{\mathbf{R}}, \ldots, y_{n+u,j}^{\mathbf{R}})$ to $\mathbf{S}$ over wire b_j and the 2-tuple $(key_{i,j}^{\mathbf{R}}, \alpha_{i,j}^{\mathbf{R}})$ to $\mathbf{S}$ over wire $b_i, 1 \leq i \leq u$, where $\alpha_{i,j}^{\mathbf{R}} = hash(key_{i,j}^{\mathbf{R}}; y_{1,j}^{\mathbf{R}}, y_{2,j}^{\mathbf{R}}, \ldots, y_{n+u,j}^{\mathbf{R}})$.

Computation by S at the end of Phase II: For $1 \leq j \leq u$, $\mathbf{S}$ receives $\beta_j^{\mathbf{S}}$ and the $n+u$-tuple $(y_{1,j}^{\mathbf{S}}, y_{2,j}^{\mathbf{S}}, \ldots, y_{n+u,j}^{\mathbf{S}})$ over wire b_j and the pair $(key_{i,j}^{\mathbf{S}}, \alpha_{i,j}^{\mathbf{S}})$ over $b_i, 1 \leq i \leq u$. $\mathbf{S}$ then does the following:

1. $\mathbf{S}$ divides the bottom band $\{b_1, b_2, \ldots, b_u\}$ into subsets $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$, where $k \leq u$, such that for any l, m, p with $1 \leq l \leq k, 1 \leq m, p \leq u$ and $b_m, b_p \in \mathcal{B}_l$, we have: (a) $\beta_m^{\mathbf{S}} = \beta_p^{\mathbf{S}}$; (b) $\alpha_{m,p}^{\mathbf{S}} = hash(key_{m,p}^{\mathbf{S}}; y_{1,p}^{\mathbf{S}}, y_{2,p}^{\mathbf{S}}, \ldots, y_{n,p}^{\mathbf{S}})$; (c) $\alpha_{p,m}^{\mathbf{S}} = hash(key_{p,m}^{\mathbf{S}}; y_{1,m}^{\mathbf{S}}, y_{2,m}^{\mathbf{S}}, \ldots, y_{p,m}^{\mathbf{S}})$.
2. For $\mathcal{B}_l$, let $b_m \in \mathcal{B}_l$ and $\beta_m^{\mathbf{S}} = \{(r_{j,l}^{\mathbf{S}}, \gamma_{j,l}^{\mathbf{S}}) : 1 \leq j \leq n\}$. $\mathbf{S}$ then computes the set

$$\mathcal{F}_l^{\mathbf{S}} = \{j : \gamma_{j,l}^{\mathbf{S}} = hash(r_{j,l}^{\mathbf{S}}; x_{1,j}^{\mathbf{S}}, x_{2,j}^{\mathbf{S}}, \ldots, x_{n+1,j}^{\mathbf{S}}), 1 \leq j \leq n\}$$

If $|\mathcal{F}_l| + |\mathcal{B}_l| \leq t$ then $\mathbf{S}$ decides that $\mathcal{B}_l$ is unacceptable set, else it is acceptable.

Before proceeding further, we make the following important claim.

Claim. Let f_i and b_j be two honest wire in top and bottom band respectively. Then at the end of **Phase II**, at least n elements in $(x_{1,i}^{\mathbf{S}}, x_{2,i}^{\mathbf{S}}, \ldots, x_{n+1,i}^{\mathbf{S}})$ and $(y_{1,j}^{\mathbf{R}}, y_{2,j}^{\mathbf{R}}, \ldots, y_{n+u,j}^{\mathbf{R}})$ are information theoretically secure.

The **Phase III** of the protocol is as follows:

Phase III: S to R: For each acceptable set $\mathcal{B}_l$ and corresponding set $\mathcal{F}_l$, **S** does the following:

1. **S** considers the first n elements from the $n+1$ tuples which it had sent over the wires in $\mathcal{F}_l$ during **Phase I** and the first n elements from the $(n+u)$ tuples which **S** had received over the wires in $\mathcal{B}_l$ during **Phase II**. By using them, **S** computes his version of n authentication keys $\mathcal{C}^{\mathbf{S}}_{1,l} = \sum_{f_j \in \mathcal{F}_l} x^{\mathbf{S}}_{1,j} + \sum_{b_j \in \mathcal{B}_l} y^{\mathbf{S}}_{1,j}$, $\mathcal{C}^{\mathbf{S}}_{2,l} = \sum_{f_j \in \mathcal{F}_l} x^{\mathbf{S}}_{2,j} + \sum_{b_j \in \mathcal{B}_l} y^{\mathbf{S}}_{2,j}$, ..., $\mathcal{C}^{\mathbf{S}}_{n,l} = \sum_{f_j \in \mathcal{F}_l} x^{\mathbf{S}}_{n,j} + \sum_{b_j \in \mathcal{B}_l} y^{\mathbf{S}}_{n,j}$.
2. For each element of $m^{\mathbf{S}}$ (recall that $|m^{\mathbf{S}}| = \frac{n}{3}$), **S** takes three elements from the keys computed in the previous step and computes the set $\mathcal{S}^{\mathbf{S}}_l = \{(c^{\mathbf{S}}_{i,l}, d^{\mathbf{S}}_{i,l}) : 1 \leq i \leq \frac{n}{3}\}$ where $(c^{\mathbf{S}}_{i,l}, d^{\mathbf{S}}_{i,l}) = USauth(m^{\mathbf{S}}_i; \mathcal{C}^{\mathbf{S}}_{3i-2}, \mathcal{C}^{\mathbf{S}}_{3i-1}, \mathcal{C}^{\mathbf{S}}_{3i}), 1 \leq i \leq \frac{n}{3}$.
3. **S** sends the set $\mathcal{F}_l, \mathcal{B}_l$ and $\mathcal{S}^{\mathbf{S}}_l$ to **R** over all the wires in the set $\mathcal{F}_l$ and terminates.

Message Recovery by R: Let **R** receives the sets $\mathcal{F}^{\mathbf{R}}_{j,l}, \mathcal{B}^{\mathbf{R}}_{j,l}$ and $\mathcal{S}^{\mathbf{R}}_{j,l}$ along wire $f_j, 1 \leq j \leq n$, for $1 \leq l \leq u$. **R** then does the following:

1. If for some $j \in \{1, 2, \ldots, n\}$ and some $l \in \{1, 2, \ldots, u\}$, $|\mathcal{F}^{\mathbf{R}}_{j,l}| + |\mathcal{B}^{\mathbf{R}}_{j,l}| \leq t$, then **R** concludes that wire f_j is corrupted and neglects all the values received along f_j.
2. If f_j is not neglected, then for each $\mathcal{F}^{\mathbf{R}}_{j,l}, \mathcal{B}^{\mathbf{R}}_{j,l}$ and $\mathcal{S}^{\mathbf{R}}_{j,l}$ received along f_j, **R** does the following: let $\mathcal{S}^{\mathbf{R}}_{j,l} = \{(c^{\mathbf{R}}_{j,i,l}, d^{\mathbf{R}}_{j,i,l}) : 1 \leq i \leq \frac{n}{3}\}$. By using the index of the wires in $\mathcal{F}^{\mathbf{R}}_{j,l}$ and $\mathcal{B}^{\mathbf{R}}_{j,l}$, **R** computes his version of authentication keys $\mathcal{C}^{\mathbf{R}}_{j,1,l}, \mathcal{C}^{\mathbf{R}}_{j,2,l}, \ldots, \mathcal{C}^{\mathbf{R}}_{j,n,l}$. Then for each $1 \leq i \leq \frac{n}{3}$, **R** applies the verification process of $USauth$ on $c^{\mathbf{R}}_{j,i,l}, d^{\mathbf{R}}_{j,i,l}, \mathcal{C}^{\mathbf{R}}_{3i-2}, \mathcal{C}^{\mathbf{R}}_{3i-1}$ and $\mathcal{C}^{\mathbf{R}}_{3i}$. If the verification is successful for all $1 \leq i \leq \frac{n}{3}$, then **R** recovers $m^{\mathbf{R}}_i$ from $c^{\mathbf{R}}_{j,i,l}, 1 \leq j \leq \frac{n}{3}$. Finally, **R** concatenates $m^{\mathbf{R}}_1, m^{\mathbf{R}}_2, \ldots, m^{\mathbf{R}}_{\frac{n}{3}}$ to reconstruct the secret $m^{\mathbf{R}}$ and terminates.

Theorem 2. *Protocol $\Pi^{existing}_{modified}$ is a three phase USMT protocol which securely sends $\Theta(n\kappa)$ bits by communicating $O(n^3\kappa)$ bits with very high probability.*

Proof: For complete proof, see [7]. □

3 Unconditionally Secure Pad Establishment Protocol

We now propose a six phase protocol called Π^{Pad}, which securely establishes a random non-zero one time pad between **S** and **R** with very high probability by communicating $O(n^3)$ field elements. If the entire bottom band is corrupted, then the size of the pad is $\Theta(n^2u)$. Otherwise the size of the pad is $\Theta(n^3)$. We first design a sub-protocol Π which is used in Π^{Pad}.

Protocol Π: Suppose **S** and **R** in advance know that full bottom band is corrupted. This implies that at most $t-u$ and at least $t+1$ wires in the top band are corrupted and honest respectively. Under this assumption, we design a sub-protocol Π, which securely establishes an information theoretic secure non-zero random one time pad of size $\Theta(n^2u)$ between **S** and **R** by communicating $O(n^3)$ field elements, with very high probability.

Let $c = n^2+t-u$. $\mathbf{S}$ selects $(t+1)\times c$ random non-zero elements from $\mathbb{F}$, denoted by $k^{\mathbf{S}}_{1,1}, k^{\mathbf{S}}_{1,2}, \ldots, k^{\mathbf{S}}_{1,c}, k^{\mathbf{S}}_{2,1}, k^{\mathbf{S}}_{2,2}, \ldots, k^{\mathbf{S}}_{2,c}, \ldots, k^{\mathbf{S}}_{t+1,1}, k^{\mathbf{S}}_{t+1,2}, \ldots, k^{\mathbf{S}}_{t+1,c}$. Now using these elements, $\mathbf{S}$ constructs an $(t+1)\times c$ matrix $A^{\mathbf{S}}$, where the $j^{th}, 1 \leq j \leq t+1$ row of $A^{\mathbf{S}}$ is $[k^{\mathbf{S}}_{j,1}\ k^{\mathbf{S}}_{j,2}\ \ldots\ k^{\mathbf{S}}_{j,i}\ \ldots\ k^{\mathbf{S}}_{j,c}]$. Now consider the $i^{th}, 1 \leq i \leq c$ column of A containing the elements $[k^{\mathbf{S}}_{1,i}\ k^{\mathbf{S}}_{2,i}\ \ldots k^{\mathbf{S}}_{t+1,i}]^T$. $\mathbf{S}$ forms a t degree polynomial $q_i(x)$ passing through the $t+1$ points $[(1, k^{\mathbf{S}}_{1,i}), (2, k^{\mathbf{S}}_{2,i}), \ldots, (t+1, k^{\mathbf{S}}_{t+1,i})]$ and evaluates $q_i(x)$ at $x = t+2, t+3, \ldots, n$ to get $y^{\mathbf{S}}_{t+2,i}, y^{\mathbf{S}}_{t+3,i}, \ldots, y^{\mathbf{S}}_{n,i}$ respectively. Finally, $\mathbf{S}$ constructs the matrix $B^{\mathbf{S}}$ of size $n \times c$, where the $i^{th}, 1 \leq i \leq c$ column of $B^{\mathbf{S}}$ is $[k^{\mathbf{S}}_{1,i}\ k^{\mathbf{S}}_{2,i}\ \ldots k^{\mathbf{S}}_{t+1,i}\ y^{\mathbf{S}}_{t+2,i}\ y^{\mathbf{S}}_{t+3,i}\ \ldots\ y^{\mathbf{S}}_{n,i}]^T$, the n points on $q_i(x)$.

Now using the $j^{th}, 1 \leq j \leq n$ row of $B^{\mathbf{S}}$, $\mathbf{S}$ forms a $n^2+t-u-1$ degree polynomial $F^{\mathbf{S}}_j(x) = k^{\mathbf{S}}_{j,1} + k^{\mathbf{S}}_{j,2}x^1 + k^{\mathbf{S}}_{j,3}x^2 + \ldots + k^{\mathbf{S}}_{j,c}x^{c-1}$. $\mathbf{S}$ also selects n random and non-zero distinct elements from $\mathbb{F}$, denoted by $\alpha^{\mathbf{S}}_1, \alpha^{\mathbf{S}}_2, \ldots, \alpha^{\mathbf{S}}_n$. Now the protocol Π is formally expressed in Table 4.

Table 4. Protocol Π

Computation and Communication by S: Along wire $f_j, 1 \leq j \leq n$, $\mathbf{S}$ sends to $\mathbf{R}$ the polynomial $F^{\mathbf{S}}_j(x)$, the random value $\alpha^{\mathbf{S}}_j$ and n tuple $[v^{\mathbf{S}}_{1j}\ v^{\mathbf{S}}_{2j}\ \ldots\ v^{\mathbf{S}}_{nj}]$ where $v^{\mathbf{S}}_{ij} = F^{\mathbf{S}}_i(\alpha^{\mathbf{S}}_j), 1 \leq i \leq n$. Let $\mathcal{V}^{\mathbf{S}}$ denotes the concatenation of the elements in the first $t+1$ rows of $B^{\mathbf{S}}$. $\mathbf{S}$ computes $\mathcal{P}^{\mathbf{S}} = \mathbf{EXTRAND}_{|\mathcal{V}^{\mathbf{S}}|,(u+1)n^2}(\mathcal{V}^{\mathbf{S}})$. The vector $\mathcal{P}^{\mathbf{S}}$ denotes the information theoretically secure random pad of size $\Theta(n^2u)$ which will be correctly established with $\mathbf{R}$ with very high probability.

Computation by R:

1. Let $\mathbf{R}$ receives $F^{\mathbf{R}}_j(x)$, the random value $\alpha^{\mathbf{R}}_j$ and the n tuple $[v^{\mathbf{R}}_{1j}\ v^{\mathbf{R}}_{2j}\ \ldots\ v^{\mathbf{R}}_{nj}]$ along wire $f_j, 1 \leq j \leq n$.
2. For $1 \leq j \leq n$, $\mathbf{R}$ computes $Support_j = |\{i : F^{\mathbf{R}}_j(\alpha^{\mathbf{R}}_i) = v^{\mathbf{R}}_{ji}\}|$. If $Support_j \geq t+1$, then $\mathbf{R}$ concludes that $F^{\mathbf{R}}_j(x)$ is a valid polynomial. Otherwise, $\mathbf{R}$ concludes that $F^{\mathbf{R}}_j(x)$ is an invalid polynomial.
3. Since there are at least $t+1$ honest wires in the top band, $\mathbf{R}$ will get at least $t+1$ valid polynomials. Now using $t+1$ valid polynomials, $\mathbf{R}$ will construct array $B^{\mathbf{R}}$. From $B^{\mathbf{R}}$, $\mathbf{R}$ computes $\mathcal{V}^{\mathbf{R}}$, from which it finally computes $\mathcal{P}^{\mathbf{R}}$ and terminates. With very high probability, $\mathcal{P}^{\mathbf{R}} = \mathcal{P}^{\mathbf{S}}$ (see Lemma 3).

Theorem 3. *If full bottom band is corrupted, then protocol Π securely establishes a random non-zero pad of $\Theta(n^2u\kappa)$ bits by communicating $O(n^3\kappa)$ bits.*

PROOF: For complete proof see [7]. □

Six Phase Protocol Π^{Pad}: We now present the protocol Π^{Pad} which uses protocols Π and $\Pi^{Existing}_{modified}$ as black-box. The first two phases of the protocol are given in Table 5. Before proceeding further, we make the following claim.

Claim. Let b_j and f_i be two honest wire in bottom and top band respectively. Then at the end of **Phase II**, at least n^2 elements in the tuple $(y^{\mathbf{R}}_{1,j}, y^{\mathbf{R}}_{2,j}, \ldots, y^{\mathbf{R}}_{n^2+1,j})$ and $(x^{\mathbf{S}}_{1,i}, x^{\mathbf{S}}_{2,i}, \ldots, x^{\mathbf{S}}_{n^2+t,i})$ are information theoretically secure.

As in protocol $\Pi^{existing}_{modified}$, from the properties of hash function, it is straightforward to check that the following holds: (a) If f_i is an honest wire in the top band and $f_i \in \mathcal{F}_l$, then with very high probability, the random (n^2+t)-tuples that $\mathbf{R}$ has received along the wires in $\mathcal{F}_l$ are not modified; (b) If f_i is an honest wire in the top band and $f_i \in \mathcal{F}_l$, then $\mathcal{F}_l$ is an acceptable set.

Table 5. First two phases of Protocol Π^{Pad}

Phase I: R to S: Corresponding to each wire $b_j, 1 \leq j \leq u$ in the bottom band, $\mathbf{R}$ selects a random non-zero n^2+1 tuple $(y^{\mathbf{R}}_{1,j}, y^{\mathbf{R}}_{2,j}, \ldots, y^{\mathbf{R}}_{n^2+1,j})$ and sends it to $\mathbf{S}$.

Phase II: S to R:

1. Let $\mathbf{S}$ receives $(y^{\mathbf{S}}_{1,j}, y^{\mathbf{S}}_{2,j}, \ldots, y^{\mathbf{S}}_{n^2+1,j})$ along wire b_j. Corresponding to each wire $b_j, 1 \leq j \leq u$, $\mathbf{S}$ selects a random non-zero hash key r_j from $\mathbb{F}$ and computes the set $\beta^{\mathbf{S}} = \{(r^{\mathbf{S}}_j, \gamma^{\mathbf{S}}_i) : 1 \leq j \leq u\}$, where $\gamma^{\mathbf{S}}_j = hash(r^{\mathbf{S}}_j; y^{\mathbf{S}}_{1,j}, y^{\mathbf{S}}_{2,j}, \ldots, y^{\mathbf{S}}_{n^2+1,j})$.
2. $\mathbf{S}$ associates a random non-zero n^2+t tuple $(x^{\mathbf{S}}_{1,j}, x^{\mathbf{S}}_{2,j}, \ldots, x^{\mathbf{S}}_{n^2+t,j})$ with wire $f_j, 1 \leq j \leq n$ in the top band. Moreover, in order to hash the tuple, $\mathbf{S}$ selects n random non-zero keys from $\mathbb{F}$ denotes by $key^{\mathbf{S}}_{i,j}$, for $1 \leq i \leq n$.
3. For each $1 \leq j \leq n$, $\mathbf{S}$ sends the set $\beta^{\mathbf{S}}$ and the (n^2+t) tuple $(x^{\mathbf{S}}_{1,j}, x^{\mathbf{S}}_{2,j}, \ldots, x^{\mathbf{S}}_{n^2+t,j}$ to $\mathbf{R}$ along wire f_j and the 2-tuple $(key^{\mathbf{S}}_{i,j}, \alpha^{\mathbf{S}}_{i,j})$ to $\mathbf{R}$ along wire $f_i, 1 \leq i \leq n$, where $\alpha^{\mathbf{S}}_{i,j} = hash(key^{\mathbf{S}}_{i,j}; x^{\mathbf{S}}_{1,j}, x^{\mathbf{S}}_{2,j}, \ldots, x^{\mathbf{S}}_{n^2+t,j})$.

Computation by R at the end of Phase II:

1. For each $1 \leq j \leq n$, $\mathbf{R}$ receives the set $\beta^{\mathbf{R}}_j$ and the (n^2+t) tuple $(x^{\mathbf{R}}_{1,j}, x^{\mathbf{R}}_{2,j}, \ldots, x^{\mathbf{R}}_{n^2+t,j})$ along wire f_j and the 2-tuple $(key^{\mathbf{R}}_{i,j}, \alpha^{\mathbf{R}}_{i,j})$ along wire $f_i, 1 \leq i \leq n$.
2. $\mathbf{R}$ divides the top band $\{f_1, f_2, \ldots, f_n\}$ into subsets $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$, where $k \leq t+1$, such that for any l, m, p with $1 \leq l \leq k, 1 \leq m, p \leq n$ and $f_m, f_p \in \mathcal{F}_l$, we have: (a) $\beta^{\mathbf{R}}_m = \beta^{\mathbf{R}}_p$; (b) $\alpha^{\mathbf{R}}_{m,p} = hash(key^{\mathbf{R}}_{m,p}; x^{\mathbf{R}}_{1,p}, x^{\mathbf{R}}_{2,p}, \ldots, x^{\mathbf{R}}_{n^2+t,p})$; (c) $\alpha^{\mathbf{R}}_{p,m} = hash(key^{\mathbf{R}}_{p,m}; x^{\mathbf{R}}_{1,m}, x^{\mathbf{R}}_{2,m}, \ldots, x^{\mathbf{R}}_{n^2+t,m})$.
3. For $\mathcal{F}_l$, let $f_m \in \mathcal{F}_l$ and $\beta^{\mathbf{R}}_m = \{(r^{\mathbf{R}}_{j,l}, \gamma^{\mathbf{R}}_{j,l}) : 1 \leq j \leq u\}$. $\mathbf{R}$ computes the set

$$\mathcal{B}_l = \{j : \gamma^{\mathbf{R}}_{j,l} = hash(r^{\mathbf{R}}_{j,l}; y^{\mathbf{R}}_{1,j}, y^{\mathbf{R}}_{2,j}, \ldots, y^{\mathbf{S}}_{n^2+1,j}), 1 \leq j \leq u\}$$

If $|\mathcal{F}_l| + |\mathcal{B}_l| \leq t$ then $\mathbf{S}$ decides that $\mathcal{F}_l$ is unacceptable, else it is acceptable set.

In the worst case, in $\mathbf{R}$'s view, there can be at most $t+1$ acceptable sets because the adversary can control at most t wires in the top band. So there can be t acceptable sets, corresponding to t corrupted wires and one acceptable set corresponding to all the honest wires in the top band. The remaining phases of the protocol is shown in Table 6.

Theorem 4. *If the entire bottom band is corrupted then Π^{Pad} securely establishes a random non-zero pad of size $\Theta(n^2\kappa)$ bits between $\mathbf{S}$ and $\mathbf{R}$ with very high probability. Otherwise, it establishes a random non-zero pad of size $\Theta(n^3\kappa)$ bits between $\mathbf{S}$ and $\mathbf{R}$ with very high probability. In either case, the protocol terminates in six phases and communicates $O(n^3\kappa)$ bits.*

Table 6. Remaining phases in Protocol Π^{Pad}

Phase III: R to S: For each acceptable set $\mathcal{F}_l$ and corresponding set $\mathcal{B}_l$, $\mathbf{R}$ does the following:

1. $\mathbf{R}$ concatenates the first n^2 elements from (n^2+1) and (n^2+t) tuples, which it had sent and received over the wires in $\mathcal{B}_l$ and $\mathcal{F}_l$ respectively. Let $\mathcal{V}_l^{\mathbf{R}}$ denotes the resultant vector.
2. Corresponding to vector $\mathcal{V}_l^{\mathbf{R}}$, $\mathbf{R}$ selects a random non-zero hash key $\mathcal{K}_l^{\mathbf{R}}$ from $\mathbb{F}$. $\mathbf{R}$ then computes the 2-tuple $(\mathcal{K}_l^{\mathbf{R}}, \gamma_l^{\mathbf{R}} = hash(\mathcal{K}_l^{\mathbf{R}}; \mathcal{V}_l^{\mathbf{R}}))$. $\mathbf{R}$ then sends $\mathcal{B}_l, \mathcal{F}_l$ and the 2-tuple $(\mathcal{K}_l^{\mathbf{R}}, \gamma_l^{\mathbf{R}})$ to $\mathbf{S}$ through all the wires in $\mathcal{B}_l$.

Computation by S at the end of Phase III: Now using the hash value(s) received from $\mathbf{R}$, $\mathbf{S}$ tries to find whether there exists at least one uncorrupted wire in the bottom band. For this, $\mathbf{S}$ does the following:

1. Let $\mathbf{S}$ receives index set $\mathcal{F}_{j,l}^{\mathbf{S}}$ and $\mathcal{B}_{j,l}^{\mathbf{S}}$ and 2-tuple $(\mathcal{K}_{j,l}^{\mathbf{S}}, \gamma_{j,l}^{\mathbf{S}})$ along wire $b_j, 1 \leq j \leq u$ for $1 \leq l \leq t+1$. If for some $j \leq u$ and some $l \leq t+1$, $|\mathcal{F}_{j,l}^{\mathbf{S}}| + |\mathcal{B}_{j,l}^{\mathbf{S}}| \leq t$, then $\mathbf{S}$ concludes that wire b_j is corrupted and neglects all the values received along b_j.
2. If $\mathcal{F}_{j,l}^{\mathbf{S}}$, $\mathcal{B}_{j,l}^{\mathbf{S}}$ and the tuple $(\mathcal{K}_{j,l}^{\mathbf{S}}, \gamma_{j,l}^{\mathbf{S}})$ is not neglected in the previous step (i.e., b_j is not discarded), then after knowing the index of the wires in $\mathcal{F}_{j,l}^{\mathbf{S}}$ and $\mathcal{B}_{j,l}^{\mathbf{S}}$, $\mathbf{S}$ computes his version of the vector $\mathcal{V}_{j,l}^{\mathbf{S}}$. Here $\mathcal{V}_{j,l}^{\mathbf{S}}$ denotes the concatenation of first n^2 values from the (n^2+1) and (n^2+t) tuples, which $\mathbf{S}$ had received and sent over the wires in $\mathcal{B}_{j,l}^{\mathbf{S}}$ and $\mathcal{F}_{j,l}^{\mathbf{S}}$ respectively. $\mathbf{S}$ now checks $\gamma_{j,l}^{\mathbf{S}} \stackrel{?}{=} hash(\mathcal{K}_{j,l}^{\mathbf{S}}; \mathcal{V}_{j,l}^{\mathbf{S}})$.
3. If the test in the last step succeeds for some $l \leq t+1$ and $j \leq u$, then $\mathbf{S}$ concludes that the tuples that are exchanged along the wires in $\mathcal{B}_{j,l}^{\mathbf{S}}$ and $\mathcal{F}_{j,l}^{\mathbf{S}}$ are correctly established between $\mathbf{S}$ and $\mathbf{R}$. $\mathbf{S}$ now applies **EXTRAND** to $\mathcal{V}_{j,l}^{\mathbf{S}}$ to generate a vector $\mathcal{P}_1^{\mathbf{S}}$ of size tn^2. Finally $\mathbf{S}$ terminates the protocol by sending a special predefined "success" value from $\mathbb{F}$, along with the index of the wires in the set $\mathcal{B}_{j,l}^{\mathbf{S}}$ and $\mathcal{F}_{j,l}^{\mathbf{S}}$ to $\mathbf{R}$ by executing the protocol $\Pi_{modified}^{existing}$. $\mathbf{R}$ securely (and hence correctly) receives these indexes with very high probability and computes his version of $\mathcal{P}_1^{\mathbf{R}}$ and terminates. Since $\Pi_{modified}^{existing}$ takes three phases, the protocol will terminate at the end of **Phase VI**.
4. If the test in step 3 fails for all l and j, then $\mathbf{S}$ concludes that entire bottom band is corrupted. In this case, $\mathbf{S}$ sends a special "failure" value from $\mathbb{F}$ to $\mathbf{R}$ by executing the three phase $\Pi_{modified}^{existing}$ protocol. Parallely, $\mathbf{S}$ establishes a secure pad $\mathcal{P}_2^{\mathbf{S}}$ of size $\Theta(n^2u)$ with $\mathbf{R}$ by executing single phase Protocol Π. At the end of $\Pi_{modified}^{existing}$, $\mathbf{R}$ will know that the entire bottom band is corrupted. Parallely at the end of Π, $\mathbf{R}$ will output $\mathcal{P}_2^{\mathbf{R}}$, with which very high probability is same as $\mathcal{P}_2^{\mathbf{S}}$. Since $\Pi_{modified}^{existing}$ takes three phases, the protocol will terminate at the end of **Phase VI**.

4 URMT with Constant Factor Overhead

Let $u \leq t$ and $n = \max(2t-u+1, t+1)$. Then we present an URMT protocol called Π^{URMT} which sends a message $m^{\mathbf{S}}$ containing ℓ field elements by communicating $O(\ell)$ field elements with very high probability, where $\ell = (t - \frac{u}{2} + 1)n^2 = \Theta(n^3)$. The total communication complexity of the protocol is $O(n^3)$ field elements and the protocol terminates in $O(u)$ phases. The principle behind the protocol is to create a win-win situation as follows: if the adversary corrupts at

most $t - \frac{u}{2}$ wires in the top band, then $\mathbf{R}$ recovers the message from the information which it receives from the honest wires in the top band. On the other hand, if more than $t - \frac{u}{2}$ wires are corrupted in the top band, then majority wires in the bottom band will be honest and so both $\mathbf{S}$ and $\mathbf{R}$ comes to know about the identity of corrupted wires in the top band by using the honest wires in the bottom band. Now using this information, $\mathbf{S}$ can re-send $m^{\mathbf{S}}$ so that $\mathbf{R}$ can recover it correctly.

As a part of pre-processing step, $\mathbf{S}$ and $\mathbf{R}$ securely establishes $\Theta(n)$ random non-zero elements from $\mathbb{F}$ with each other in advance with very high probability by executing the three phase protocol $\Pi^{existing}_{modified}$. Let the set of these elements be denoted by $\mathcal{K}$. The elements in $\mathcal{K}$ will be used by $\mathbf{S}$ and $\mathbf{R}$ as authentication and hash keys to reliably exchange the outcome of certain steps during the execution of the protocol Π^{URMT}. Note that elements in $\mathcal{K}$ need not be distinct, but they are randomly selected from $\mathbb{F}$. We assume that initially all the elements in $\mathcal{K}$ are marked as "unused". Each time $\mathbf{S}$ ($\mathbf{R}$) needs a key(s) for hashing or authentication, then the first "unused" element(s) from $\mathcal{K}$ is/are selected as key(s). In order to do the verification, $\mathbf{R}$ ($\mathbf{S}$) also uses the same element(s) from $\mathcal{K}$ as keys. Once the verification is done, the element(s) is/are marked as "used" Thus we can view $\mathcal{K}$ as a global set, which is parallely used by both $\mathbf{S}$ and $\mathbf{R}$.

Let $m^{\mathbf{S}} = [m^{\mathbf{S}}_{1,1}\ m^{\mathbf{S}}_{1,2}\ \ldots\ m^{\mathbf{S}}_{1,n^2}\ m^{\mathbf{S}}_{2,1}\ m^{\mathbf{S}}_{2,2}\ \ldots\ m^{\mathbf{S}}_{2,n^2}\ \ldots\ m^{\mathbf{S}}_{t-\frac{u}{2}+1,1}\ m^{\mathbf{S}}_{t-\frac{u}{2}+1,2}\ \ldots\ m^{\mathbf{S}}_{t-\frac{u}{2}+1,n^2}]$ be the message. $\mathbf{S}$ constructs array $B^{\mathbf{S}}$ of size $n \times n^2$ from $m^{\mathbf{S}}$ in same way as in protocol Π with following modifications: $\mathbf{S}$ first constructs the array $A^{\mathbf{S}}$ of size $(t - \frac{u}{2} + 1) \times n^2$ from $m^{\mathbf{S}}$, where the $j^{th}, 1 \leq j \leq (t - \frac{u}{2} + 1)$ row of $A^{\mathbf{S}}$ is $[m^{\mathbf{S}}_{j,1}\ m^{\mathbf{S}}_{j,2}\ \ldots\ m^{\mathbf{S}}_{j,n^2}]$. By considering the elements in individual columns as distinct points, $\mathbf{S}$ interpolates the unique $(t - \frac{u}{2})$ degree polynomial passing through them. $\mathbf{S}$ then further evaluates the interpolated polynomials at additional $(t - \frac{u}{2})$ values of x and gets the array $B^{\mathbf{S}}$. Now by considering the elements along $j^{th}, 1 \leq j \leq n$ row of $B^{\mathbf{S}}$ as coefficients, $\mathbf{S}$ constructs $F^{\mathbf{S}}_j(x)$ of degree $n^2 - 1$. First two phases of Π^{URMT} are as follows:

Phase I: S to R: Along wire $f_j, 1 \leq j \leq n$, $\mathbf{S}$ sends to $\mathbf{R}$ the polynomial $F^{\mathbf{S}}_j(x)$, a random non-zero value $\alpha^{\mathbf{S}}_j$ and n tuple $[v^{\mathbf{S}}_{1j}\ v^{\mathbf{S}}_{2j}\ \ldots\ v^{\mathbf{S}}_{nj}]$ where $v^{\mathbf{S}}_{ij} = F^{\mathbf{S}}_i(\alpha^{\mathbf{S}}_j), 1 \leq i \leq n$.

Phase II: R to S

1. Let $\mathbf{R}$ receives $F^{\mathbf{R}}_j(x)$, the value $\alpha^{\mathbf{R}}_j$ and the n tuple $[v^{\mathbf{R}}_{1j}\ v^{\mathbf{R}}_{2j}\ \ldots\ v^{\mathbf{R}}_{nj}]$ along wire $f_j, 1 \leq j \leq n$.
2. For $1 \leq j \leq n$, $\mathbf{R}$ computes $Support_j = |\{i : F^{\mathbf{R}}_j(\alpha^{\mathbf{R}}_i) = v^{\mathbf{R}}_{ji}\}|$. Let $\mathcal{P}^{\mathbf{R}}$ denotes the set of wires f_j, such that $Support_j \geq (t - \frac{u}{2} + 1)$. In addition, $\mathbf{R}$ constructs a directed graph $\mathcal{G}^{\mathbf{R}} = (\mathcal{V}^{\mathbf{R}}, \mathcal{E}^{\mathbf{R}})$, called *conflict graph*, where $\mathcal{V}^{\mathbf{R}} = \{f_1, f_2, \ldots, f_n\}$ and arc $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$ if $F^{\mathbf{R}}_i(\alpha^{\mathbf{R}}_j) \neq v^{\mathbf{R}}_{ij}$.
3. Corresponding to graph $\mathcal{G}^{\mathbf{R}}$, $\mathbf{R}$ constructs a conflict list $\mathcal{Y}^{\mathbf{R}}$ of five tuples where for each arc $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$, there exists a five tuple $(f_i, f_j, \alpha^{\mathbf{R}}_j, F^{\mathbf{R}}_i(\alpha^{\mathbf{R}}_j), v^{\mathbf{R}}_{ij})$ in $\mathcal{Y}^{\mathbf{R}}$. $\mathbf{R}$ sends $\mathcal{Y}^{\mathbf{R}}$ to $\mathbf{S}$ through bottom band.

Before proceeding further, we make the following claim.

Claim. Let f_i be a wire which has delivered incorrect $F_i^{\mathbf{R}}(x) \neq F_i^{\mathbf{S}}(x)$ to $\mathbf{R}$ and f_j be an honest wire. Then with very high probability $(f_i, f_j) \in \mathcal{E}^{\mathbf{R}}$.

PROOF: For complete proof see [7]. □

Now $\mathbf{S}$ considers the conflict list which it receives identically through at least $\frac{u}{2}+1$ wires. If $\mathbf{S}$ does not receives any conflict list identically through at least $\frac{u}{2}+1$ wires, then $\mathbf{S}$ concludes that at least $\frac{u}{2}+1$ wires are corrupted in the bottom band, which further implies that at most $t-\frac{u}{2}-1$ wires are corrupted in the top band. In this case, the protocol proceeds as shown in Table 7.

Table 7. Execution of Π^{URMT} if $\mathbf{S}$ does not receives $\frac{u}{2}+1$ identical conflict lists

Phase III: S to R: By selecting two elements from $\mathcal{K}$ as authentication keys, $\mathbf{S}$ authenticates an unique special predetermined signal "terminate" and sends to $\mathbf{R}$. $\mathbf{R}$ receives the signal correctly with very high probability and concludes that at most $t-\frac{u}{2}$ wires have delivered incorrect values during **Phase I**. So by using the polynomials received along the first $t-\frac{u}{2}+1$ wires in $\mathcal{P}^{\mathbf{R}}$ during **Phase I**, $\mathbf{R}$ constructs the array $B^{\mathbf{R}}$. From $B^{\mathbf{R}}$, $\mathbf{R}$ recovers $m^{\mathbf{R}}$ and terminates.

Lemma 3. *If* $\mathbf{S}$ *does not receives the same conflict list through at least* $\frac{u}{2}+1$ *wires then with very high probability,* $\mathbf{R}$ *correctly recovers* $m^{\mathbf{S}}$ *from the polynomials delivered by the wires in* $\mathcal{P}^{\mathbf{R}}$.

PROOF: For complete proof see [7]. □

If at the end of **Phase III**, $\mathbf{S}$ receives the same conflict list, say $\mathcal{Y}^{\mathbf{S}}$ through at least $\frac{u}{2}+1$ wires, then $\mathbf{S}$ does the following: let the five tuples in $\mathcal{Y}^{\mathbf{S}}$ be of the form $(f_i, f_j, \alpha_j'^{\mathbf{R}}, F_i'^{\mathbf{R}}(\alpha_j'^{\mathbf{R}}), v_{ij}'^{\mathbf{R}})$. For each such four tuple, $\mathbf{S}$ checks $\alpha_j'^{\mathbf{R}} \stackrel{?}{=} \alpha_j^{\mathbf{S}}$ and $v_{ij}^{\mathbf{S}} \stackrel{?}{=} v_{ij}'^{\mathbf{R}}$. If any of these test fails then $\mathbf{S}$ concludes that wire f_j has delivered incorrect values to $\mathbf{R}$ during **Phase I** and adds f_j to a list $L_{fault}^{\mathbf{S}}$. On the other hand, if both the test passes then $\mathbf{S}$ checks $F_i^{\mathbf{S}}(\alpha_j^{\mathbf{S}}) \stackrel{?}{=} F_i'^{\mathbf{R}}(\alpha_j'^{\mathbf{R}})$. If the test fails then $\mathbf{S}$ concludes that wire f_i has delivered incorrect $F_i'^{\mathbf{R}}(x) \neq F_i^{\mathbf{S}}(x)$ to $\mathbf{R}$ during **Phase I** and adds f_i to $L_{fault}^{\mathbf{S}}$. Note that $\mathbf{S}$ does not know whether $\mathcal{Y}^{\mathbf{S}}$ is a genuine conflict list and is indeed sent by $\mathbf{R}$. But still $\mathbf{S}$ computes $L_{fault}^{\mathbf{S}}$.

$\mathbf{S}$ now finds the cardinality of list $L_{fault}^{\mathbf{S}}$. Now there are two possible cases. If $|L_{fault}^{\mathbf{S}}| \leq (t-\frac{u}{2})$, then $\mathbf{S}$ concludes that at least $t-\frac{u}{2}+1$ wires have delivered correct polynomial during **Phase I**. $\mathbf{S}$ then performs the same computation as shown in Table 7. The correctness of the protocol in this execution sequence is given by Lemma 4.

Lemma 4. *If* $|L_{fault}^{\mathbf{S}}| \leq (t-\frac{u}{2})$, *then with very high probability,* $\mathbf{R}$ *can correctly recover* $m^{\mathbf{S}}$ *from the polynomials delivered by the wires in* $\mathcal{P}^{\mathbf{R}}$.

PROOF: For complete proof see [7]. □

If $|L_{fault}^{\mathbf{S}}| \geq (t-\frac{u}{2}+1)$, then $\mathbf{S}$ further communicates with $\mathbf{R}$ to find whether $\mathcal{Y}^{\mathbf{S}}$ was indeed sent by $\mathbf{R}$. For this, $\mathbf{S}$ and $\mathbf{R}$ executes the steps in Table 8.

Before proceeding further, we state the following lemma.

Table 8. Execution of Π^{URMT} if $|L^{\mathbf{S}}_{fault}| \geq (t - \frac{u}{2} + 1)$

Phase III: S to R: $\mathbf{S}$ selects $2|L^{\mathbf{S}}_{fault}|$ elements from the set $\mathcal{K}$ as authentications keys and using them authenticates each element of $L^{\mathbf{S}}_{fault}$ by using $URauth$ function. Let $L^{\mathbf{S}}_{fault_{auth}}$ denotes the set of corresponding authenticated values. $\mathbf{S}$ then sends $(\mathcal{Y}^{\mathbf{S}}, L^{\mathbf{S}}_{fault}, L^{\mathbf{S}}_{fault_{auth}})$ to $\mathbf{R}$ through top band.

Phase IV: R to S: Let $\mathbf{R}$ receives $(\mathcal{Y}^{\mathbf{R}}_j, L^{\mathbf{R}}_{fault_j}, L^{\mathbf{R}}_{fault_{j,auth}})$ from $\mathbf{S}$ along wire $f_j, 1 \leq j \leq n$. From these values, $\mathbf{R}$ now tries to find out whether $\mathbf{S}$ has correctly received the original $\mathcal{Y}^{\mathbf{R}}$ over more that $\frac{u}{2} + 1$ wires during **Phase I**, and if yes, then the corresponding $L^{\mathbf{S}}_{fault}$. For this, $\mathbf{R}$ does the following:

1. For each $1 \leq j \leq n$, $\mathbf{R}$ checks $\mathcal{Y}^{\mathbf{R}}_j \stackrel{?}{=} \mathcal{Y}^{\mathbf{R}}$ and $|L^{\mathbf{R}}_{fault_j}| \geq (t - \frac{u}{2} + 1)$. In any of the test fails, then $\mathbf{R}$ neglects all the values received along f_j. Otherwise, $\mathbf{R}$ applies the $URauth$ function to each element of $L^{\mathbf{R}}_{fault_j}$ by using the same keys from $\mathcal{K}$, which were used by $\mathbf{S}$ to authenticate $L^{\mathbf{S}}_{fault}$ and computes the set $L'^{\mathbf{R}}_{fault_{j,auth}}$. $\mathbf{R}$ then checks $L'^{\mathbf{R}}_{fault_{j,auth}} \stackrel{?}{=} L^{\mathbf{R}}_{fault_{j,auth}}$. If the test fails then again $\mathbf{R}$ discards the values received along f_j.
2. If as a result of previous step, $\mathbf{R}$ has discarded the values along all the wires in the top band, then $\mathbf{R}$ concludes that $\mathbf{S}$ has not received original $\mathcal{Y}^{\mathbf{R}}$ over more that $\frac{u}{2} + 1$ wires during **Phase I**, which further implies that at most $t - \frac{u}{2} - 1$ wires were corrupted in the top band during **Phase I**. So $\mathbf{R}$ recovers $m^{\mathbf{R}}$ by using the polynomials received over the first $t - \frac{u}{2} + 1$ wires in $\mathcal{P}^{\mathbf{R}}$ during **Phase I**. Moreover, by selecting next two "unused" elements k_1, k_2 from $\mathcal{K}$ as authentication keys, $\mathbf{R}$ computes $response_1 = URauth(\text{"}terminate\text{"}; k_1, k_2)$ where "terminate" is an unique pre-defined special element from $\mathbb{F}$. $\mathbf{R}$ then send the tuple $(\text{"}terminate\text{"}, response_1)$ to $\mathbf{S}$ through the bottom band and terminates.
3. If during step 1, there exists a $j \in \{1, 2, \ldots, n\}$ such that $\mathcal{Y}^{\mathbf{R}}_j = \mathcal{Y}^{\mathbf{R}}$, $|L^{\mathbf{R}}_{fault_j}| \geq (t - \frac{u}{2} + 1)$ and $L'^{\mathbf{R}}_{fault_{j,auth}} = L^{\mathbf{R}}_{fault_{j,auth}}$, then $\mathbf{R}$ concludes that $\mathbf{S}$ has correctly received original $\mathcal{Y}^{\mathbf{R}}$ over more that $\frac{u}{2} + 1$ wires during **Phase I** and $L^{\mathbf{R}}_{fault_j}$ is the corresponding L_{fault} sent by $\mathbf{S}$. So $\mathbf{R}$ removes the wires in $L^{\mathbf{R}}_{fault_j}$ from his view for further computation and communication. Note that if there are more than one such j (whose probability is negligible), then $\mathbf{R}$ arbitrarily selects one. Now by selecting k_1, k_2 from $\mathcal{K}$ as authentication keys, $\mathbf{R}$ computes $response_2 = URauth(\text{"}continue\text{"}; k_1, k_2)$ where "continue" is an unique pre-defined special element from $\mathbb{F}$. $\mathbf{R}$ then send the tuple $(\text{"}continue\text{"}, response_2)$ to $\mathbf{S}$ through the bottom band.

Computation by S at the end of Phase IV: $\mathbf{S}$ checks whether it is getting any 2-tuple identically over at least $\frac{u}{2} + 1$ wires. If not, then $\mathbf{S}$ concludes that $\mathbf{R}$ has recovered $m^{\mathbf{R}}$ and terminates. On the other hand, if $\mathbf{S}$ receives a 2-tuple say $(x^{\mathbf{S}}_1, y^{\mathbf{S}}_1)$ over $\frac{u}{2} + 1$ wires, then $\mathbf{S}$ verifies $y^{\mathbf{S}}_1 \stackrel{?}{=} URauth(x^{\mathbf{S}}_1; k_1, k_2)$. If the test fails, then $\mathbf{S}$ again concludes that $\mathbf{R}$ has recovered $m^{\mathbf{R}}$ and terminates. On the other hand, if the test succeeds then $\mathbf{S}$ further checks $x^{\mathbf{S}}_1 \stackrel{?}{=} \text{"}terminate\text{"}$. If yes, then $\mathbf{S}$ again concludes that $\mathbf{R}$ has recovered $m^{\mathbf{R}}$ and terminates. If no then $\mathbf{S}$ concludes that $\mathcal{Y}^{\mathbf{S}}$ was indeed sent by $\mathbf{R}$.

Lemma 5. *If* $|L^{\mathbf{S}}_{fault}| \geq (t - \frac{u}{2} + 1)$*, then at the end of* **Phase III** *in Table 8 one of the following will happen:*

1. *If* $\mathbf{S}$ *has "not" received the original* $\mathcal{Y}^{\mathbf{R}}$ *over more that* $\frac{u}{2} + 1$ *wires during* **Phase I***, then with very high probability* $\mathbf{R}$ *will be able to detect this. Moreover* $\mathbf{R}$ *will be able to correctly recover* $m^{\mathbf{R}}$ *by using the polynomials received over the wires in* $\mathcal{P}^{\mathbf{R}}$ *with very high probability.*
2. *If* $\mathbf{S}$ *has received the original* $\mathcal{Y}^{\mathbf{R}}$ *over more that* $\frac{u}{2} + 1$ *wires during* **Phase I***, then* $\mathbf{R}$ *will be able to detect this. Moreover, with very high probability,* $\mathbf{R}$ *will correctly receive* $L^{\mathbf{S}}_{fault}$*, from which it will come to know the identity of at least* $|L^{\mathbf{S}}_{fault}|$ *corrupted wires in the top band.*

Table 9. Execution of Π^{URMT} to re-send $m^{\mathbf{S}}$

$\mathbf{S}$ divides $m^{\mathbf{S}}$ into blocks $B^{\mathbf{S}}_1, B^{\mathbf{S}}_2, \ldots, B^{\mathbf{S}}_{\frac{u}{2}}$, each of size $\frac{|m^{\mathbf{S}}|}{\frac{u}{2}}$. Moreover $\mathbf{S}$ and $\mathbf{R}$ initializes variables $wc^{\mathbf{S}} = 1, bc^{\mathbf{S}} = 1$ and $wc^{\mathbf{R}} = 1, bc^{\mathbf{R}} = 1$ respectively. $\mathbf{S}$ and $\mathbf{R}$ now executes the following steps:

1. While $(wc^{\mathbf{S}} \leq \frac{u}{2} - 1)$ and (all the blocks of $m^{\mathbf{S}}$ are not sent) do
   (a) $\mathbf{S}$ sends the block $B^{\mathbf{S}}_{bc^{\mathbf{S}}}$ to $\mathbf{R}$ *only* over wire $f_{wc^{\mathbf{S}}}$ in the top band.
   (b) Let $\mathbf{R}$ receives $B^{\mathbf{R}}_{bc^{\mathbf{R}}}$ along wire $f_{wc^{\mathbf{R}}}$. Now by selecting k_{bc} from the set $\mathcal{K}$ as hash key, $\mathbf{R}$ computes $x^{\mathbf{R}}_{bc} = hash(k_{bc}; B^{\mathbf{R}}_{bc^{\mathbf{R}}})$ and sends $x^{\mathbf{R}}_{bc}$ to $\mathbf{S}$ through the bottom band.
   (c) $\mathbf{S}$ correctly receives $x^{\mathbf{R}}_{bc}$ through at least $\frac{u}{2} + 1$ wires (recall that in this case majority wires in bottom band are honest) and verifies $x^{\mathbf{R}}_{bc} \stackrel{?}{=} hash(k_{bc}; B^{\mathbf{S}}_{bc^{\mathbf{S}}})$. If the test fails then $\mathbf{S}$ concludes that wire $f_{wc^{\mathbf{S}}}$ has delivered incorrect $B^{\mathbf{S}}_{bc^{\mathbf{S}}}$ to $\mathbf{R}$. So $\mathbf{S}$ increments $wc^{\mathbf{S}}$ by one. Moreover, $\mathbf{S}$ authenticates an unique pre-defined special "increment-wire" element from $\mathbb{F}$ by using two keys from the set $\mathcal{K}$ and sends it to $\mathbf{R}$ through the top band. $\mathbf{R}$ correctly receives the signal with very high probability and accordingly increments $wc^{\mathbf{R}}$ by one.
   On the other hand, if the test succeeds then $\mathbf{S}$ concludes that wire $f_{wc^{\mathbf{S}}}$ has delivered correct $B^{\mathbf{S}}_{bc^{\mathbf{S}}}$ to $\mathbf{R}$. So $\mathbf{S}$ increments $bc^{\mathbf{S}}$ by one. Moreover, $\mathbf{S}$ authenticates an unique pre-defined special "increment-block" value from $\mathbb{F}$ by using two keys from the set $\mathcal{K}$ and sends it to $\mathbf{R}$ through the top band. $\mathbf{R}$ correctly receives the signal with very high probability and accordingly increments $bc^{\mathbf{R}}$ by one.
2. If all the blocks of $m^{\mathbf{S}}$ are sent then both $\mathbf{S}$ and $\mathbf{R}$ terminates. Otherwise $\mathbf{S}$ concatenates all the remaining blocks of $m^{\mathbf{S}}$ and sends to $\mathbf{R}$ through wire $f_{\frac{u}{2}}$ and terminates. $\mathbf{R}$ correctly receives these blocks and terminates.

If at the end of **Phase IV** in Table 8, $\mathbf{S}$ recovers "continue" signal from $\mathbf{R}$ then $\mathbf{S}$ removes the wires in $L^{\mathbf{S}}_{fault}$ from his view. $\mathbf{S}$ now knows that in both $\mathbf{S}$ and $\mathbf{R}$'s view, there are $n - |L^{\mathbf{S}}_{fault}|$ wires in the top band, of which at most $t - |L^{\mathbf{S}}_{fault}|$ could be corrupted. Since $|L^{\mathbf{S}}_{fault}| \geq (t - \frac{u}{2} + 1)$, in $\mathbf{S}$ and $\mathbf{R}$'s view, there are at most $t - \frac{u}{2}$ wires in the top band, of which at most $\frac{u}{2} - 1$ could be corrupted. Moreover, both $\mathbf{S}$ and $\mathbf{R}$ now knows that there exists at least $\frac{u}{2} + 1$ honest wires in the bottom band. $\mathbf{S}$ now proceeds to re-send $m^{\mathbf{S}}$. For this, out of

the $t - \frac{u}{2}$ in their view, both $\mathbf{S}$ and $\mathbf{R}$ considers only the first $\frac{u}{2}$ wires. Without loss of generality, let these be the wires $f_1, f_2, \ldots, f_{\frac{u}{2}}$. Now both $\mathbf{S}$ and $\mathbf{R}$ knows that at least one wire among these $\frac{u}{2}$ wires is honest. $\mathbf{S}$ now re-sends $m^{\mathbf{S}}$ by executing the steps given in Table 9. This will take $\Theta(u)$ phases.

Lemma 6. *If $\mathcal{Y}^{\mathbf{R}}$ is correctly received by $\mathbf{S}$ over more than $\frac{u}{2}+1$ wires during* **Phase II** *and if the corresponding $|L^{\mathbf{S}}_{fault}| \geq (t - \frac{u}{2} + 1)$, then with very high probability, $\mathbf{S}$ will be able to correctly re-send $m^{\mathbf{S}}$ by executing the steps in Table 9 by incurring a communication overhead of $O(|m^{\mathbf{S}}|)$ field elements.*

Theorem 5. *If $m^{\mathbf{S}}$ is a message containing ℓ field elements where $\ell \geq (t - \frac{u}{2} + 1)n^2$, then there exists an $O(u)$ phase URMT protocol which reliably sends $m^{\mathbf{S}}$ with very high probability by communicating $O(\ell)$ field elements. In terms of bits, the protocol sends $\ell\kappa$ bits by communicating $O(\ell\kappa)$ bits.*

Remark 1. In Π^{URMT}, we assumed that $u \leq t$. If $u > t$, then we can modify the protocol to reliably send a message containing $(\frac{u}{2}+1)n^2 = \Theta(n^3)$ field elements with a communication overhead of $O(n^3)$ field elements.

5 Communication Optimal USMT Protocol

We now design an $O(u)$ phase USMT protocol called Π^{USMT}, which sends a message $\mathbf{M}^{\mathbf{S}}$ containing ℓ field elements by communicating $O(n^3)$ field elements with very high probability. If the full bottom band is corrupted then $\ell = \Theta(n^2u)$, otherwise $\ell = \Theta(n^3)$. The protocol is as follows:

1. Depending upon whether the full bottom band is corrupted or not, $\mathbf{S}$ and $\mathbf{R}$ securely establishes a random non-zero one time pad *Pad* of length $\Theta(n^2u)$ or $\Theta(n^3)$ with very high probability by executing the protocol Π^{Pad}.
2. If *Pad* is of length $\Theta(n^2u)$, then $\mathbf{S}$ selects a secret message $\mathbf{M}^{\mathbf{S}}$ of length $\Theta(n^2u)$. $\mathbf{S}$ then computes $C = \mathbf{M}^{\mathbf{S}} \oplus Pad$ and reliably sends C to $\mathbf{R}$ with very high probability by executing the protocol Π^{URMT}. $\mathbf{R}$ correctly receives C with very high probability and recovers $\mathbf{M}^{\mathbf{R}} = C \oplus Pad$. On the other hand, if *Pad* is of length $\Theta(n^3)$, then $\mathbf{S}$ and $\mathbf{R}$ does the same computation, except that $\mathbf{M}^{\mathbf{S}}$ and C (and hence $\mathbf{M}^{\mathbf{R}}$) will be of length $\Theta(n^3)$.

6 Lower Bound on the Communication Complexity

An obvious lower bound on communication complexity of URMT protocols to send a message containing ℓ field elements is $\Omega(\ell)$. Since, we have already shown that this bound is tight by designing Π^{URMT}, we need not have to prove the lower bound for URMT. Similarly, if at least one wire in the bottom band is uncorrupted, then $\Omega(\ell)$ is a trivial lower bound on the communication complexity of any USMT protocol to securely send ℓ field elements. Again, since we have already shown that this bound is tight by designing Π^{USMT} (which securely sends ℓ field elements by communicating ℓ field elements if there exists at least one uncorrupted wire in the bottom band), we need not have to prove the lower bound for this case. The lower bound for remaining case is given by Theorem 6.

Theorem 6. *Suppose there exists $u \leq t$ wires in the bottom band and $n = max(2t - u + 1, t + 1)$ wires in the top band. Moreover, the entire bottom band is corrupted. Then any multiphase USMT protocol to send a message $M^{\mathbf{S}}$ containing ℓ field elements from $\mathbb{F}$, needs to communicate $\Omega(\frac{n\ell}{u})$ field elements. In terms of bits, the protocol needs to communicate $\Omega(\frac{n\ell}{u}\kappa)$ bits to send $\ell\kappa$ bits.*

PROOF: The lower bound is derived by using entropy based arguments. We do not give the proof here due to space constraint. For complete proof see [7]. The lower bound in Theorem 6 is tight. Specifically, if the entire bottom band is corrupted, then protocol Π^{USMT} satisfies the lower bound. □

7 Conclusion and Open Problems

In this paper we have designed communication optimal URMT and USMT protocol in directed networks, which are first of their kind. It would be interesting to reduce the phase complexity of our URMT and USMT protocols. Our URMT and USMT protocols are communication optimal in amortized sense; i.e., they achieve bit optimality for sufficiently large message size (ℓ). We leave the issue of designing communication optimal URMT/USMT protocols in directed networks for small sized message as an open problem.

References

1. Beerliová-Trubíniová, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 305–328. Springer, Heidelberg (2006)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10 (1988)
3. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Proc. of FOCS 1988, pp. 11–19 (1988)
4. Desmedt, Y., Wang, Y.: Perfectly secure message transmission revisited. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002)
5. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. JACM 40(1), 17–47 (1993)
6. Franklin, M., Wright, R.: Secure communication in minimal connectivity models. Journal of Cryptology 13(1), 9–30 (2000)
7. Patra, A., Choudhary, A., Pandu Rangan, C.: Unconditionally reliable and secure message transmission in directed networks revisited. Cryptology ePrint Archive, Report 2008/262
8. Patra, A., Choudhary, A., Srinathan, K., Pandu Rangan, C.: Unconditionally reliable and secure message transmission in undirected synchronous networks: Possibility, feasibility and optimality. Cryptology ePrint Archive, Report 2008/141
9. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC, pp. 73–85 (1989)
10. Shanker, B., Gopal, P., Srinathan, K., Pandu Rangan, C.: Unconditional reliable message transmision in directed networks. In: Proc. of SODA 2008 (2008)

11. Srinathan, K., Narayanan, A., Rangan, C.P.: Optimal perfectly secure message transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)
12. Wang, Y., Desmedt, Y.: Perfectly secure message transmission revisited (manuscript), `www.sis.uncc.edu/yonwang/`
13. Yao, A.C.: Protocols for secure computations. In: Proc. of 23rd IEEE FOCS, pp. 160–164 (1982)

Linear Bandwidth Naccache-Stern Encryption

Benoît Chevallier-Mames[1], David Naccache[2], and Jacques Stern[2]

[1] DCSSI, Laboratoire de cryptographie,
51, Boulevard de la Tour Maubourg
75700 Paris, France
benoit.chevallier-mames@sgdn.gouv.fr
[2] École normale supérieure, Équipe de cryptographie,
45 rue d'Ulm, F-75230 Paris CEDEX 05, France
{david.naccache,jacques.stern}@ens.fr

Abstract. The Naccache-Stern (NS) knapsack cryptosystem is an original yet little-known public-key encryption scheme. In this scheme, the ciphertext is obtained by multiplying public-keys indexed by the message bits modulo a prime p. The cleartext is recovered by factoring the ciphertext raised to a secret power modulo p.

NS encryption requires a multiplication per two plaintext bits on the average. Decryption is roughly as costly as an RSA decryption. However, NS features a bandwidth sublinear in $\log p$, namely $\log p / \log\log p$. As an example, for a 2048-bit prime p, NS encryption features a 233-bit bandwidth for a 59-kilobyte public key size.

This paper presents new NS variants achieving bandwidths *linear* in $\log p$. As linear bandwidth claims a public-key of size $\log^3 p / \log\log p$, we recommend to combine our scheme with other bandwidth optimization techniques presented here.

For a 2048-bit prime p, we obtain figures such as 169-bit plaintext for a 10-kilobyte public key, 255-bit plaintext for a 20-kilobyte public key or a 781-bit plaintext for a 512-kilobyte public key. Encryption and decryption remain unaffected by our optimizations: As an example, the 781-bit variant requires 152 multiplications per encryption.

Keywords: Public key cryptography, NS cryptosystem, multiplicative knapsack, efficiency.

1 Introduction

The Naccache-Stern cryptosystem (NS), introduced a decade ago in [NS97], is a public-key cryptosystem based on the following problem:

$$\text{given } p,\ c \text{ and a set } \{p_i\}, \text{ find a binary vector } x \text{ such that } c = \prod_{i=0}^{n-1} p_i^{x_i} \bmod p.$$

Trivially, if the p_i-s are relatively prime and much smaller than p, the above problem can be solved in polynomial time by factoring c in $\mathbb{N}$.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 327–339, 2008.

A trapdoor is obtained by extracting a secret (s-th) modular root of each p_i and publishing these roots, denoted $v_i = \sqrt[s]{p_i} \mod p$. By raising a product of such roots to the s-th power, each v_i shrinks back to a much smaller p_i and x can be found by factoring the result in $\mathbb{N}$.

Unfortunately, no security proofs linking NS's security to standard complexity assumptions are known, but at the same time, no efficient chosen-plaintext attacks against NS's one-wayness are known either.

More formally, let p be a large public prime[1] and denote by n the largest integer such that:

$$p > \prod_{i=0}^{n-1} p_i \text{ where } p_i \text{ is the } i\text{-th prime (start from } p_0 = 2).$$

The secret-key $0 < s < (p-1)$ is a random integer such that $\gcd(p-1, s) = 1$ and the public-keys are the n roots:

$$v_i = \sqrt[s]{p_i} \bmod p.$$

A message $m = \sum_{i=0}^{n-1} 2^i m_i$, where $m_i \in \{0, 1\}$, is encrypted as $c = \prod_{i=0}^{n-1} v_i^{m_i} \bmod p$ and recovered by:

$$m = \sum_{i=0}^{n-1} \frac{2^i}{p_i - 1} \times \Big(\gcd(p_i, c^s \bmod p) - 1 \Big).$$

Denoting by $\ln(x)$ natural logarithms and by $\log(x)$ base-2 logarithms, it is easy to see that NS's bandwidth is sublinear: As $p_i \sim i \ln i$, we have

$$\ln p \sim \sum_{i=0}^{n} \ln p_i \sim n \ln n \;\Rightarrow\; \ln \ln p \sim \ln n,$$

which in turn gives:

$$n \sim \frac{\ln p}{\ln \ln p} \sim \frac{\log p}{\log \log p}.$$

In a typical setting, a 2048-bit p corresponds to a sixteen kilobyte public-key and allows encrypting 233-bit messages.

[NS97] also describes a variant depending on a parameter $\ell \in \mathbb{N}$. Here, p is such that

$$p > \prod_{i=0}^{n-1} {p_i}^{\ell}.$$

$m = \sum_{i=0}^{n-1} (\ell+1)^i m_i$, expressed in base $(\ell+1)$ (here $m_i \in [0, \ell]$), is encrypted as

$$c = \prod_{i=0}^{n-1} v_i^{m_i} \bmod p,$$

[1] For technical reasons, p must be a safe prime, *cf.* to Sections 2.4 of [NS97] or 5.

and decryption is straightforwardly modified. In this paper, we refer this version as the "$(\ell+1)$-base variant".

The goal of this work is to improve the scheme's bandwidth using more sophisticated arithmetic encoding approaches. Indeed, 233-bit plaintexts are often insufficiently large in practice to include the message, the randomizer (typically 128 bits) and the redundancy (at least 160 bits, or better 256 bits) that one needs to use chosen-ciphertext secure transformations such Fujisaki and Okamoto's [FO99, FO00].

In the next section, we propose a technique based on modular fractions that multiplies bandwidth by $\log_2 3 \simeq 1.58$ for binary-message NS.[2] Section 3 describes a new message encoding technique that dramatically increases bandwidth (to become linear in $\log p$). Section 4 extends the previous idea to $(\ell+1)$-base NS, thereby further increasing bandwidth. Final figures are given in Section 4.3. In Section 5, we examine the security of the proposed improvements. Finally, Section 6 describes combinatorial problems whose solutions might yield even more efficient NS variants.

2 Fractional Message Encoding

In this section, we show that using signed message bits allows to increase bandwidth at no cost. Consider a message represented in a signed binary notation, *i.e.*,

$$m = \sum_{i=0}^{n-1} 2^i\, m_i \quad \text{where } m_i \in \{-1, 0, 1\}$$

and an unchanged encryption procedure. During decryption, the receiver recovers a u such that:

$$u = c^s = \frac{a}{b} \mod p, \qquad \text{with} \begin{cases} a = \prod_{m_i=1} p_i \\ b = \prod_{m_i=-1} p_i. \end{cases} \quad \text{and } \gcd(a,b) = 1.$$

The following theorem shows that, given u, one can recover a and b efficiently using Gauss's algorithm for finding the shortest vector in a two-dimensional lattice [Val91].

Theorem 1 ([FSW02]). *Let $a, b \in \mathbb{Z}$ such that $|a| \leq A$ and $0 < b \leq B$. Let p be a prime such that $2AB < p$. Let $u = a/b \mod p$. Then given $\{A, B, u, p\}$, one can recover $\{a, b\}$ in polynomial time.*

Taking $A = B = \lfloor \sqrt{p} \rfloor - 1$, we have that $2AB < p$. If we assume in addition that m was such that $0 \leq a \leq A$ and $0 < b \leq B$, we can recover a and b from s in polynomial time. And by testing the divisibility of a and b by the small primes p_i, the receiver can eventually recover m as before.

[2] This factor becomes $\log_{1+\ell}(2\ell+1)$ for the $(\ell+1)$-base variant.

But what happens if $|a| > A$ or $b > B$?

To tackle this case too, let us tweak the definition of p to $p > 2^w \times \prod_{i=0}^{n-1} p_i$, for some small integer $w \geq 1$ (we suggest to take $w = 50$), and define a finite sequence $\{A_i, B_i\}$ of integers such that:

$$A_i = 2^{wi} \quad \text{and} \quad B_i = \left\lfloor \frac{p-1}{2A_i} \right\rfloor .$$

For all $i > 0$, we have that $ab < 2A_iB_i < p$. Moreover, there must exist at least one index i such that $0 \leq a \leq A_i$ and $0 < b \leq B_i$. Then using the algorithm of Theorem 1, given A_i, B_i, p and s, one can recover a and b, and eventually recover m. The problem is that we have just lost the guarantee that such an $\{a, b\}$ is unique. Namely, we could in theory hit another $\{a', b'\}$ whose modular ratio gives the same u, for some other index $i' \neq i$. But we expect this to happen with negligible probability for large enough w.

Senders wishing to eliminate even the negligible probability that decryption will produce more than one plaintext can still simulate the decryption's Gaussian phase and, in case, re-randomize m until decryption becomes unambiguous.

The effect of this optimization is noteworthy as for the price of a constant increase (*e.g.* $\simeq 50$ bits) in p, bandwidth is multiplied by a factor of $\log_2 3$.

It is important to underline that while different signed binary representations of a given m exist (*e.g.* $10\underline{1} = 011$ *i.e.* $2^2 - 2^0 = 2^1 + 2^0$), the above procedure will recover the *exact* encoding used to encrypt m and not an equivalent one.

Note that as $\ell > 1$ is used in conjunction with this technique (*i.e.,* in the $(\ell + 1)$-base variant), bandwidth improvement tends to one bit per prime as ℓ grows. Namely, fractional encoding increases bandwidth from $n \log(1 + \ell)$ to $n \log(1 + 2\ell)$.

3 Small Prime Packing

Let the integer $\gamma \geq 2$ be a system parameter. We now group the small primes p_i into n packs containing γ small primes each.[3] That is, the first pack will contain primes p_1 to p_γ, the second pack will include primes $p_{\gamma+1}$ to $p_{2\gamma}$ *etc.* As previously, the p_i-s are indexed in increasing order.

We also update the condition on the large prime p to:

$$\prod_{i=1}^{n} p_{\gamma i} < p.$$

In other words, we do not request p to be larger than the product of all the small primes. Instead, we *only* request p to be larger than the product of the largest representatives of each pack.

[3] For the sake of simplicity, we now define the first prime as $p_1 = 2$.

We now represent m in base γ, *i.e.*,

$$m = \sum_{i=0}^{n-1} \gamma^i \, m_i \quad \text{where} \quad m_i \in [0, \gamma - 1]$$

and encode m by picking in pack i the prime representing the message's i-th digit m_i and multiplying all so chosen p_i-s modulo p:

$$\mathsf{encoding}(m) = \prod_{i=0}^{n-1} p_{\gamma i + m_i + 1} \bmod p.$$

We can now apply this encoding to the NS and re-define encryption as:

$$c = \mathsf{encryption}(m) = \sqrt[s]{\mathsf{encoding}(m)} = \prod_{i=0}^{n-1} v_{\gamma i + m_i + 1} \bmod p.$$

To decrypt c, the receiver computes $u = c^s \bmod p$ and recovers m by factoring u. Note that as soon as a representative of pack i is found, the receiver can stop sieving within pack i and start decrypting digit $i + 1$.

3.1 A Small Example

We illustrate the mechanism by a small toy-example.

• KEY GENERATION FOR $n = 3$ AND $\gamma = 4$:

The prime $p = 4931 > p_\gamma \times p_{2\gamma} \times p_{3\gamma} = 7 \times 19 \times 37$ and the secret $s = 3079$ yield the v-list:

$$\text{pack 1}\begin{cases} v_1 = \sqrt[s]{2} \bmod p = 1370 \\ v_2 = \sqrt[s]{3} \bmod p = 1204 \\ v_3 = \sqrt[s]{5} \bmod p = 1455 \\ v_4 = \sqrt[s]{7} \bmod p = 3234 \end{cases} \quad \text{pack 2}\begin{cases} v_5 = \sqrt[s]{11} \bmod p = 2544 \\ v_6 = \sqrt[s]{13} \bmod p = 3366 \\ v_7 = \sqrt[s]{17} \bmod p = 1994 \\ v_8 = \sqrt[s]{19} \bmod p = 3327 \end{cases} \quad \text{pack 3}\begin{cases} v_9 = \sqrt[s]{23} \bmod p = 4376 \\ v_{10} = \sqrt[s]{29} \bmod p = 1921 \\ v_{11} = \sqrt[s]{31} \bmod p = 3537 \\ v_{12} = \sqrt[s]{37} \bmod p = 3747 \end{cases}$$

• ENCRYPTION OF $m = 35$:

We start by writing m is base $\gamma = 4$, *i.e.*, $m = 35 = 203_4$ and encrypt it as:

$$c = v_{(0 \cdot 4 + \mathbf{3} + 1)} \times v_{(1 \cdot 4 + \mathbf{0} + 1)} \times v_{(2 \cdot 4 + \mathbf{2} + 1)} = v_4 \times v_5 \times v_{11} \bmod 4931 = 4484.$$

• DECRYPTION:

By exponentiation, the receiver retrieves:

$$c^s \bmod p = 4484^{3079} \bmod 4931 = 7 \times 11 \times 31 = p_{(0 \cdot 4 + \mathbf{3} + 1)} \times p_{(1 \cdot 4 + \mathbf{0} + 1)} \times p_{(2 \cdot 4 + \mathbf{2} + 1)},$$

whereby $m = 203_4$.

3.2 Bandwidth Considerations

The bandwidth gain stems from the fact that, for large i, we have $p_{\gamma i+1} \simeq p_{\gamma i+\gamma}$ which allows the new format to accommodate $\log_2 \gamma$ message bits at the price of one single $p_{\gamma i+\gamma}$. This situation is much more favorable than the original NS, where each message bit costs a new p_i.

More precisely, $p_{\gamma i} \sim \gamma i \ln i$ yields an $(n \log \gamma)$-bit bandwidth where:

$$n \sim \ln p / \ln \ln p \sim \log p / \log \log p$$

The bandwidth gain is thus a constant multiplicative factor (namely $\log \gamma$) and the increase in n is logarithmic. Note that at the same time, the v_i-list becomes γ times longer.

The following table shows the performances of the new encoding algorithm for a 2048-bit p. The first row represents the original NS for the sake of comparison.

γ	n	plaintext bits $= n \log \gamma$	public key size $= \gamma n \log p$	information rate$= n \frac{\log \gamma}{\log p}$
NS	233	233 bits	59 kilobytes	0.11
2	208	207 bits	104 kilobytes	0.10
4	189	378 bits	189 kilobytes	0.18
8	172	516 bits	344 kilobytes	0.25
16	159	635 bits	636 kilobytes	0.31
32	147	734 bits	1176 kilobytes	0.36
64	137	821 bits	2192 kilobytes	0.40
128	128	896 bits	4096 kilobytes	0.44
256	121	967 bits	7744 kilobytes	0.47
512	114	1025 bits	14592 kilobytes	0.50
1024	108	1080 bits	27648 kilobytes	0.53

As one can see, bandwidth improvement is significant, but the public keys are huge. Fortunately, we address this issue in the Section 4.

3.3 Linear Bandwidth

Setting $\gamma = n$ (*i.e.,* n packs containing n primes each), we can approximate:

$$p_{\gamma i} \sim \gamma i \ln i \sim n i \ln i \;\Rightarrow\; \sum_{i=0}^{n} \ln p_{\gamma i} \sim \sum_{i=0}^{n} \ln(n^2 \ln n) \sim n \ln(n^2) \sim 2n \ln n \sim \ln p.$$

As $\ln n \sim \ln \ln p$, we get an $n \log n$ bit bandwidth with:

$$n \sim \frac{\ln p}{2 \ln \ln p} \sim \frac{\log p}{2 \log \log p}.$$

Substituting the expressions of n and $\log n$ into the bandwidth formula (that is $n \log n$), we see that the resulting information rate turns out to be $\frac{1}{2}$. This encoding scheme therefore features a linear bandwidth, while NS is only sublinear. Note that this format is compatible with fractional encoding (Section 2), thereby allowing further constant-factor bandwidth gains.

3.4 Optimizing the Encoding of Zeros

We now observe that the encoding of zeros does not require using new primes. The corresponding tweak to the encryption procedure is straightforward and allows to lower the number of p_i-s from γn to $(\gamma - 1)n$. This increases n and hence the information rate.

For the previous toy-example, the packs will become:

$$\text{pack 1} \begin{cases} v_1 = 1 \\ v_2 = \sqrt[s]{2} \bmod p \\ v_3 = \sqrt[s]{3} \bmod p \\ v_4 = \sqrt[s]{5} \bmod p \end{cases} \quad \text{pack 2} \begin{cases} v_5 = 1 \\ v_6 = \sqrt[s]{7} \bmod p \\ v_7 = \sqrt[s]{11} \bmod p \\ v_8 = \sqrt[s]{13} \bmod p \end{cases} \quad \text{pack 3} \begin{cases} v_9 = 1 \\ v_{10} = \sqrt[s]{17} \bmod p \\ v_{11} = \sqrt[s]{19} \bmod p \\ v_{12} = \sqrt[s]{23} \bmod p \end{cases}$$

p can now be chosen as $p = 1499 > p_\gamma \times p_{2\gamma} \times p_{3\gamma} = 5 \times 13 \times 23$, which is indeed somewhat shorter than the modulus used in Section 3.1.

Figures are given in the following table, where the first row (*i.e.*, $\gamma = 2$) represents the original NS. As before, this results assume a 2048-bit p. The optimization is particularly interesting for small γ values.

γ	n	plaintext bits $= n \log \gamma$	public key size $= (\gamma - 1)\, n \log p$	information rate$= n \frac{\log \gamma}{\log p}$
2	233	233 bits	59 kilobytes	0.11 (original NS)
4	196	392 bits	147 kilobytes	0.19
8	175	525 bits	307 kilobytes	0.26
16	160	640 bits	600 kilobytes	0.31
32	148	740 bits	1147 kilobytes	0.36
64	137	822 bits	2158 kilobytes	0.40
128	128	896 bits	4064 kilobytes	0.44
256	121	968 bits	7714 kilobytes	0.47
512	114	1026 bits	14564 kilobytes	0.50
1024	108	1080 bits	27621 kilobytes	0.53

4 Using Powers of Primes

In this section we apply prime-packing to the $(\ell+1)$-base variant. We start with an example, to explain as simply as possible the obtained scheme.

4.1 A Small Example

Take $n = 1$ and $\gamma = 4$, *i.e.* a single pack, containing $\{p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7\}$. We also set $\ell = 2$, pick a modulus $p > 7^\ell = 7^2 = 49$, define the public key as:

$$\{v_1 = \sqrt[s]{2} \bmod p, v_2 = \sqrt[s]{3} \bmod p, v_3 = \sqrt[s]{5} \bmod p, v_4 = \sqrt[s]{7} \bmod p\}$$

and consider all p_i products of weight smaller or equal to ℓ:

$$\begin{array}{lll} 7^0 \times 5^0 \times 3^0 \times 2^0 & 7^0 \times 5^0 \times 3^0 \times 2^1 & 7^0 \times 5^0 \times 3^0 \times 2^2 \\ 7^0 \times 5^0 \times 3^1 \times 2^0 & 7^0 \times 5^0 \times 3^1 \times 2^1 & 7^0 \times 5^0 \times 3^2 \times 2^0 \\ 7^0 \times 5^1 \times 3^0 \times 2^0 & 7^0 \times 5^1 \times 3^0 \times 2^1 & 7^0 \times 5^1 \times 3^1 \times 2^0 \\ 7^0 \times 5^2 \times 3^0 \times 2^0 & 7^1 \times 5^0 \times 3^0 \times 2^0 & 7^1 \times 5^0 \times 3^0 \times 2^1 \\ 7^1 \times 5^0 \times 3^1 \times 2^0 & 7^1 \times 5^1 \times 3^0 \times 2^0 & 7^2 \times 5^0 \times 3^0 \times 2^0 \end{array}$$

All in all, we have $1 + 4 + 10 = \binom{\gamma+\ell}{\ell} = 15$ products[4] that can be associated to 15 message digit values. Therefore, to encode a message digit $m_0 \in [0, 14]$, we use any unranking algorithm [SW86] returning $\mathsf{unrank}(m_0) = \{a, b, c, d\}$ and encrypt m_0 as:

$$c = \mathsf{encryption}(m_0) = {v_1}^a \times {v_2}^b \times {v_3}^c \times {v_4}^d \bmod p.$$

For instance, using a lexicographic ranking of words of weight two:

$$\begin{array}{lcl}
\mathsf{unrank}(\ 0) = \{0,0,0,0\} & \rightsquigarrow & 7^0 \times 5^0 \times 3^0 \times 2^0 \\
\mathsf{unrank}(\ 1) = \{0,0,0,1\} & \rightsquigarrow & 7^0 \times 5^0 \times 3^0 \times 2^1 \\
\mathsf{unrank}(\ 2) = \{0,0,0,2\} & \rightsquigarrow & 7^0 \times 5^0 \times 3^0 \times 2^2 \\
\mathsf{unrank}(\ 3) = \{0,0,1,0\} & \rightsquigarrow & 7^0 \times 5^0 \times 3^1 \times 2^0 \\
\mathsf{unrank}(\ 4) = \{0,0,1,1\} & \rightsquigarrow & 7^0 \times 5^0 \times 3^1 \times 2^1 \\
\mathsf{unrank}(\ 5) = \{0,0,2,0\} & \rightsquigarrow & 7^0 \times 5^0 \times 3^2 \times 2^0 \\
\mathsf{unrank}(\ 6) = \{0,1,0,0\} & \rightsquigarrow & 7^0 \times 5^1 \times 3^0 \times 2^0 \\
\mathsf{unrank}(\ 7) = \{0,1,0,1\} & \rightsquigarrow & 7^0 \times 5^1 \times 3^0 \times 2^1 \\
\mathsf{unrank}(\ 8) = \{0,1,1,0\} & \rightsquigarrow & 7^0 \times 5^1 \times 3^1 \times 2^0 \\
\mathsf{unrank}(\ 9) = \{0,2,0,0\} & \rightsquigarrow & 7^0 \times 5^2 \times 3^0 \times 2^0 \\
\mathsf{unrank}(10) = \{1,0,0,0\} & \rightsquigarrow & 7^1 \times 5^0 \times 3^0 \times 2^0 \\
\mathsf{unrank}(11) = \{1,0,0,1\} & \rightsquigarrow & 7^1 \times 5^0 \times 3^0 \times 2^1 \\
\mathsf{unrank}(12) = \{1,0,1,0\} & \rightsquigarrow & 7^1 \times 5^0 \times 3^1 \times 2^0 \\
\mathsf{unrank}(13) = \{1,1,0,0\} & \rightsquigarrow & 7^1 \times 5^1 \times 3^0 \times 2^0 \\
\mathsf{unrank}(14) = \{2,0,0,0\} & \rightsquigarrow & 7^2 \times 5^0 \times 3^0 \times 2^0
\end{array}$$

$m_0 = 12$ will be encrypted as $\mathsf{encryption}(12) = \sqrt[5]{3} \times \sqrt[5]{7} = v_2 \times v_4 \bmod p$. Decryption recovers $2^0 \times 3^1 \times 5^0 \times 7^1$ by exponentiation and determines that $m_0 = \mathsf{rank}(\{1, 0, 1, 0\}) = 12$.

The bandwidth improvement stems from the fact that we encrypt $\log(15)$ bits where the $(\ell + 1)$-base variant only encrypts $\log(3)$. In other words, the prime-packing idea fits particularly well to the $(\ell + 1)$-base system.

Also, as is all practical instances $\binom{\gamma+\ell}{\ell}$ will remain moderate (typically less than one hundred), functions $\mathsf{rank}(\cdot)$ and $\mathsf{unrank}(\cdot)$ can be implemented as simple lookup tables rather than as full-fledged constructive combinatorial algorithms.

4.2 Formal Description

Let us describe now the scheme formally. Let $\ell \geq 1$ and γ be two integer parameters[5] and consider n packs containing γ small primes each (the primes start from $p_1 = 2$). We pick a prime p such that:

$$\prod_{i=1}^{n} {p_{\gamma i}}^{\ell} < p.$$

[4] The attentive reader would rightly note that there are actually more p_i products smaller than p. This is true for very small primes in the first packs, but when one considers packs whose minimal and maximal p_i-s are roughly equivalent in size, the number of products quickly tends to $\binom{\gamma+\ell}{\ell}$.

[5] NS corresponds to the case $\{\gamma, \ell\} = \{1, 1\}$ and the $(\ell + 1)$-base variant corresponds to $\gamma = 1$.

As there are[6] shows that there are $\binom{\gamma+\ell}{\ell}$ different γ-tuples $\{d_1, \ldots, d_\gamma\}$ such that $0 \leq d_k$ and $\sum_k d_k \leq \ell$, we define $\mathsf{unrank}(\cdot)$ as an invertible function mapping integers in $[0, \binom{\gamma+\ell}{\ell} - 1]$ to $\{d_1, \ldots, d_\gamma\}$-tuples.

To encrypt a message expressed in base $\binom{\gamma+\ell}{\ell}$, *i.e.*, $m = \sum_{i=0}^{n-1} \binom{\gamma+\ell}{\ell}^i m_i$ with $m_i \in [0, \binom{\gamma+\ell}{\ell} - 1]$, one computes:

$$c = \mathsf{encryption}(m) = \prod_{i=0}^{n-1} \prod_{j=1}^{\gamma} v_{\gamma i+j}{}^{d_{i,j}} \bmod p$$

where $\{d_{i,1}, \ldots, d_{i,\gamma}\} = \mathsf{unrank}(m_i)$.

To decrypt c, the receiver simply factorizes $c^s \bmod p$ in $\mathbb{N}$ and recovers each m_i by:

$$m_i = \mathsf{rank}(\{d_{i,1}, \ldots, d_{i,\gamma}\}).$$

4.3 Bandwidth Considerations

The table below shows that the variant described in this section features a better bandwidth and smaller public-keys than the basic prime-packs encoding of Section 3. Data was generated for several public-key sizes (namely 10, 20, 50, and 500 kilobytes) and a 2048-bit p. The first line $\{\gamma, \ell\} = \{1, 1\}$ is the original NS:

γ	ℓ	n	plaintext bits $= n \log \binom{\gamma+\ell}{\ell}$	public key size $= \gamma n \log p$	information rate$= n\frac{\log \binom{\gamma+\ell}{\ell}}{\log p}$
1	1	233	233 bits	59 kilobytes	0.11
8	66	5	169 bits	10 kilobytes	0.08
16	54	5	255 bits	20 kilobytes	0.12
64	73	3	398 bits	48 kilobytes	0.19
512	38	4	781 bits	512 kilobytes	0.38
128	10	16	781 bits	512 kilobytes	0.38

Note that encryption is very fast, since it requires $\ell \cdot n$ multiplications *e.g.* in the 781-bit setting an encryption claims 152 multiplications.

5 Security Considerations

As stressed previously, no security proof is known for the original NS, and we have no hope nor claim that our modifications may supplement this lack. In this section we nonetheless recall certain security-related facts, some of which are already known since [NS97], for the clarity and the self-containment of this paper.

[6] *Cf.* to Appendix A for a proof.

5.1 What Security Can Be Attained?

The most basic security property expected from any encryption scheme is *one-wayness* (OW): an attacker should not be able to recover the plaintext given a ciphertext. We capture this notion more formally by saying that for any adversary $\mathcal{A}$, success in inverting the effect of encryption must occur with negligible probability.

Semantic Security (IND) [GM84], also known as *indistinguishability of encryptions* captures a stronger privacy notion. The adversary $\mathcal{A}$ is said to break IND when, after choosing two same-length messages m_0 and m_1, he can decide whether a given ciphertext corresponds to m_0 or to m_1. An encryption scheme is said to be semantically secure (or indistinguishable) if no probabilistic algorithm can break IND.

The original NS cryptosystem, or the variants presented in Sections 2, 3 or 4 can not ensure indistinguishability, since they are by nature deterministic. The hope however is that there might be one-way. To achieve full-security with our variants (or with NS), one can use generic transformations such as [FO99, FO00]: nevertheless, as there are no formal reductions from a standard hard problem to an attack of NS-type schemes (be these the original NS or the variants proposed herein), the application of these generic rules cannot possibly achieve a provably security, but only give empirical security arguments.

5.2 Security Arguments

Our schemes can be broken if one solves the discrete-logarithm. It is clear that a discrete-logarithm oracle will totally break the NS scheme or the variants presented in this paper. Indeed, to this aim, it is sufficient to ask the oracle for the discrete-logarithm of p_1 in base v_1, which is actually the secret key s. Even if the primes are permuted or made secret, the fact that primes must be small makes them easily guessable.

Larger message Space may Make ns-type Problems Harder. As one can see, the schemes presented in this paper are — as is the original NS — multiplicative knapsacks. Even if no efficient algorithm solving this problem is known, one must ensure that a brute-force attack consisting in testing all products is impossible. More precisely, getting back the arguments put forward in Section 2.3 of [NS97], the message space must at least exceed 160 bits, if one requires an 80-bit security, or 256 bits, if one wants 128-bit security. In this perspective, our bandwidth improvements indirectly improve security by easing the attainment of a larger message space. However, we cannot claim that the variants are stronger, as bandwidth improvements come along with larger public-keys, which — at least in the information theoretic sense — give more information to the attacker about the secret key.

Small Factors of $(p-1)$. As stressed in Section 2.4 of [NS97], the small factors of $(p-1)$ are important. Denote by $\mathbb{QR}_p$ and $\overline{\mathbb{QR}}_p$ the quadratic and non-quadratic residues modulo p respectively. Let

$$c = \prod_{i=0}^{n-1} v_i^{m_i} \bmod p.$$

By computing $a = c^{\frac{p-1}{2}} \bmod p$, one gets

$$a = \prod_{v_i \in \overline{\mathbb{QR}}_p} (-1)^{m_i} \bmod p,$$

which in turn leaks the value $\sum_{v_i \in \overline{\mathbb{QR}}_p} m_i \bmod 2$. This partial information leakage can also be applied to other small factors of $(p-1)$. Therefore, [NS97] advised to use a strong prime p, and to spare one m_i to compensate the leakage (in other words, they simply make $\sum_{v_i \in \overline{\mathbb{QR}}_p} m_i \bmod 2$ constant).

In our variants, it is not as simple to use same attacks. Indeed, in any given prime pack, one expects to have some primes in $\mathbb{QR}_p$ and others in $\overline{\mathbb{QR}}_p$. For example, with the variant of Section 3, getting $c = \mathsf{encryption}(m) = \prod_{i=0}^{n-1} v_{\gamma i + m_i + 1} \bmod p$, the attacker may compute $a = c^{\frac{p-1}{2}} \bmod p$. As

$$a = \prod_{v_{\gamma i + m_i + 1} \in \overline{\mathbb{QR}}_p} (-1) \bmod p,$$

this reveals the parity of the number of message digits m_i whose corresponding primes are in $\overline{\mathbb{QR}}_p$. Even if leakage is less precise than in the NS case, we still recommend the use of a strong prime (and residue value compensation) with our variants.

Can a Reduction from Attacking NS to Attacking our Variants Exist?. At a first glance, it might seem that reductions between NS and our variants exist: indeed, one may hope that access to a decryption oracle $\mathcal{D}$ of one of our schemes would yield an NS decryption oracle $\mathcal{D}'$.

However, a simple observation shows that this is certainly impossible: in our case, the public key is longer and contains more elements related to the secret key. Therefore, from an NS public key and a challenge, it may certainly be possible to build a challenge for our variants, but there is little hope that one might reconstruct the entire public key.

Thus, we have no formal proof that the security of the original NS is equivalent to the security of the variants proposed herein.

6 Further Research

We conclude this paper with a couple of interesting combinatorial problems whose solution might further improve the NS's bandwidth.

Setting $\ell = 1$, not all collections of γn integers allow encoding γ^n combinations. Let $\mathcal{S} = \{S_1, ..., S_n\}$ be n integer-sets, each of size γ and denote by $S_i[j]$ the j-th element of S_i. We call $\mathcal{S}$ an *encoder* if its S_i-s can be used as a collection of packs encoding exactly $n \log_2 \gamma$ bits, or, in other words, if no collisions in the

integer sub-products of $\mathcal{S}$ occur. Improving the NS consists in finding "better" encoders.

To compare encoders, we use their *head-products*, namely:

$$h(\mathcal{S}) = \prod_{i=1}^{n} \max_{j} (S_i[j])$$

Head-products lower-bound the modulus p and hence "measure" bandwidth.

We saw that when the $S_i[j]$ are the first small primes, $\mathcal{S}$ is an encoder and $h(\mathcal{S}) = \prod_{i=1}^{n} p_{i\gamma}$ (Section 3). We also saw that when the smallest element in each S_i is one, the resulting $\mathcal{S}$ is still an encoder whose head-product is $h(\mathcal{S}) = \prod_{i=1}^{n} p_{i(\gamma-1)}$ (Section 3.4).

This gives raise to interesting combinatorial problems such as finding algorithms for efficiently testing that a given $\mathcal{S}$ is an encoder, or finding algorithms for constructing optimal encoders, *i.e.* encoders featuring a minimal head-product (and consequently a maximal bandwidth).

As an example, a (rather inefficient) computer-aided exploration for $n = 3$ and $\gamma = 4$ discovered the optimal encoder $\mathcal{S}$ whose $h(\mathcal{S}) = 4 \times 8 \times 13 = 416$:

$$\text{pack 1} \begin{cases} S_1[1] = 1 \\ S_1[2] = 2 \\ S_1[3] = 3 \\ S_1[4] = 4 \end{cases} \quad \text{pack 2} \begin{cases} S_2[1] = 1 \\ S_2[2] = 5 \\ S_2[3] = 7 \\ S_2[4] = 8 \end{cases} \quad \text{pack 3} \begin{cases} S_3[1] = 1 \\ S_3[2] = 9 \\ S_3[3] = 11 \\ S_3[4] = 13 \end{cases}$$

Interestingly, this encoder contains primes, but also powers of primes. Moreover, throughout our search, non-optimal encoders containing composite integers (such as 6) were found as well.

Decoding messages encoded with such complicated $\mathcal{S}$-s might not always be straightforward as in such atypical encoders, decoding is based on impossibilities of certain factor combinations rather than on the occurrence of certain factors in the product.

The above questions also generalize to packs of rationals.

References

[FO99] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)

[FO00] Fujisaki, E., Okamoto, T.: How to enhance the security of public-key encryption at minimum cost. IEICE Transaction of Fundamentals of Electronic Communications and Computer Science E83-A(1), 24–32 (2000)

[FSW02] Fouque, P.-A., Stern, J., Wackers, J.-G.: Cryptocomputing with rationals. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 136–146. Springer, Heidelberg (2003)

[GM84] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)

[NS97] Naccache, D., Stern, J.: A new public-key cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. Springer, Heidelberg (1997)

[SW86] Stanton, D., White, D.: Constructive combinatorics. Springer, New York (1986)
[Val91] Vallée, B.: Gauss' algorithm revisited. Journal of Algorithms 12(4), 556–572 (1991)

A Computing $\mathcal{R}_{\ell,\gamma}$

In this appendix, we recall how we evaluate the number, denoted $\mathcal{R}_{\ell,\gamma}$, of different γ-tuples $\{d_1, \ldots, d_\gamma\}$ such that $0 \leq d_k$ and $\sum_k d_k \leq \ell$.

$\binom{\gamma+i-1}{i}$ is the number of sequences of γ integers whose sum equals i. Therefore, we have:

$$\mathcal{R}_{\ell,\gamma} = \sum_{i=0}^{\ell} \binom{\gamma+i-1}{i}.$$

Assume that we have $\mathcal{R}_{\ell,\gamma} = \binom{\gamma+\ell}{\ell}$. What happens for $(\ell+1)$?

$$\mathcal{R}_{\ell+1,\gamma} = \sum_{i=0}^{\ell+1} \binom{\gamma+i-1}{i} = \mathcal{R}_{\ell,\gamma} + \binom{\gamma+\ell+1-1}{\ell+1} = \binom{\gamma+\ell}{\ell} + \binom{\gamma+\ell}{\ell+1} = \binom{\gamma+\ell+1}{\ell+1}$$

where the last line stems from Pascal's rule.

As $\mathcal{R}_{0,\gamma} = 1 = \binom{\gamma}{0}$, we get by induction that:

$$\mathcal{R}_{\ell,\gamma} = \binom{\gamma+\ell}{\ell}.$$

Immunising CBC Mode Against Padding Oracle Attacks: A Formal Security Treatment

Kenneth G. Paterson and Gaven J. Watson*

Information Security Group,
Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, U.K.
kenny.paterson@rhul.ac.uk, g.watson@rhul.ac.uk

Abstract. Padding oracle attacks against CBC mode encryption were introduced by Vaudenay. They are a powerful class of side-channel, plaintext recovering attacks which have been shown to work in practice against CBC mode when it is implemented in specific ways in software. In particular, padding oracle attacks have been demonstrated for certain implementations of SSL/TLS and IPsec. In this paper, we extend the theory of provable security for symmetric encryption to incorporate padding oracle attacks. We develop new security models and proofs for CBC mode (with padding) in the chosen-plaintext setting. These models show how to select padding schemes which provably provide a strong security notion (indistinguishability of encryptions) in the face of padding oracle attacks. We also show that an existing padding method, OZ-PAD, that is recommended for use with CBC mode in ISO/IEC 10116:2006, provably resists Vaudenay's original attack, even though it does not attain our indistinguishability notion.

1 Introduction

When constructing cryptographic protocols to be used for protecting network traffic, cryptographic primitives such as block ciphers (often operating in Cipher Block Chaining (CBC) mode) and MACs are often combined in order to obtain both confidentiality and integrity of the traffic. Typical examples of such protocols are the ESP protocol in IPsec and the SSL/TLS Record Layer protocol. Theoretical analysis of such protocols until now has considered idealised cryptographic components that are always correctly implemented. Unfortunately, this is not always the case in practice, and often the exact manner in which these components are implemented is highly significant for security.

Side channel attacks are a powerful class of techniques that can be used to break implementations of cryptographic primitives which from a theoretical perspective appear to be secure. These attacks are mostly directed against embedded system and smart card deployments of cryptography, but they can also be

* This author is supported by an EPSRC Industrial CASE studentship sponsored by BT Research Laboratories.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 340–357, 2008.

used to perform attacks against network protocols. The padding oracle attack, as introduced by Vaudenay [11], is a side channel attack of this type. A padding oracle attack works against the CBC mode of operation of a block cipher. It exploits the fact that, in real implementations, data typically must be padded according to some rule before it can be encrypted. In contrast, all current theoretical security models for CBC mode assume that data is already neatly aligned on a block boundary.

In such an attack, the adversary is assumed to be equipped with a padding oracle that, given a ciphertext, tells the adversary whether or not the underlying plaintext is correctly padded according to some specific padding rule. In his original paper, Vaudenay showed that, for certain widely used padding schemes, a padding oracle could be exploited to build a decryption oracle, i.e. to recover plaintext. Further theoretical analysis of padding oracle and related attacks appeared in [3,10,12,9]. Notably, the work in [10,12] influenced the subsequent development of the ISO standard for CBC mode encryption (ISO/IEC 10116:2006), which eventually recommended a padding scheme called OZ-PAD because it appears to resist padding oracle attacks.

Canvel *et al.* [5] described a real-world example of a padding oracle attack type against the SSL/TLS Record Layer protocol as implemented in OpenSSL. The padding oracle was present because the SSL/TLS specification uses a "MAC-then-encrypt" construction and because the OpenSSL implementation behaved differently when faced with a padding failure or a MAC failure. Interestingly, the particular combination of CBC mode encryption and MAC algorithm used in SSL/TLS was proven secure by Krawczyk in [8], but his proof does not directly consider any issues arising from padding. Thus the subsequent attack by Canvel *et al.* demonstrates the limitations of existing security analysis in practice. More recently, Degabriele and Paterson [6] used an extension of the basic padding oracle techniques to find attacks against certain encryption-only configurations of IPsec, both as specified in the relevant RFC [7] and as implemented in RFC-compliant software. Again, the particular use of CBC mode in IPsec is provably secure against chosen plaintext attackers in the usual model for security for symmetric encryption [1]; however, the way in which [7] recommends dealing with padding failures actually leads to an attacker being able to construct a padding oracle for RFC-compliant implementations. Thus, simply rewriting code to eliminate padding oracles may not always be possible.

The practical attacks of [5] and [6] demonstrate the inadequacy of current security models for CBC mode encryption in the face of padding oracle attacks and provide a strong motivation for the current paper.

1.1 Our Contribution

In this paper, we extend the existing security models for CBC mode encryption [1] to include the real-world security concern that is represented by padding oracle attacks. We give results relating these different security models, in the spirit of [1]. This gives us a framework within which we can analyse the security of

specific padding schemes operating in conjunction with CBC mode. In particular, we show that the OZ-PAD padding scheme offers a form of one-way security against chosen-plaintext attack that is sufficient to resist Vaudenay's original plaintext-recovering padding oracle attack. We also show that certain padding schemes, such as the Arbitrary Tail padding methods Abit and Abyte recommended by Black and Urtubia in [3], offer much stronger security guarantees in the form of indistinguishability of encryptions against chosen-plaintext attackers equipped with padding oracles.

In this paper, for reasons of brevity, we focus exclusively on the case where the attacker is a chosen-plaintext one. The equally important case of security of authenticated encryption schemes making use of CBC mode encryption as a component will be considered in a forthcoming companion paper. There, we assume the attacker is equipped with a decryption oracle that may produce outputs allowing the adversary to distinguish between padding failures and other types of decryption failure. Such behaviour underlies the successful attacks against OpenSSL in [5], for example.

1.2 Related Work

Bellare *et al.* [1] provided the first systematic study of security notions for symmetric encryption, as well as specific security results for various modes of operation of a block cipher. Our work in this paper relies extensively on this foundational paper.

In addition to the practice-oriented work on SSL/TLS [5] and IPsec [6], several real-world security protocols have been subjected to formal security analysis. This includes the work of Bellare *et al.* [2], who extensively analysed the authenticated encryption scheme used in SSH, showing that SSH is vulnerable to a certain plaintext-guessing "reaction attack" in situations where fixed padding is used (rather than the randomised padding recommended in the SSH specification). This work also considered provable security properties of the specific encoding and cryptographic operations carried out by SSH, with the analysis explicitly taking padding into account. However, it should be noted that the security model used in [2] effectively denies the adversary access to the decryption oracle as soon as he submits an invalid ciphertext to it. This reflects what SSH does in practice, but severely limits the use that a theoretical adversary might make of such a decryption oracle. A second example of formal analysis of real-world protocols is provided by the work of Boldyreva and Kumar [4], who provided a provable security treatment of authenticated encryption in the Kerberos protocol suite. This analysis seems to take into account every component used in the Kerberos authenticated encryption algorithm *except* for padding. A third example is provided by [8], in which Krawczyk studies the security properties of the "MAC-then-encrypt" paradigm as used in SSL/TLS (but his proof does not take into account any padding).

A useful survey of side channel attacks, including padding oracle attacks, can be found in [13].

2 Definitions and Models for Chosen Plaintext Security

2.1 CBC Mode for Arbitrary Length Messages

Let $F = \{F_K : K \in \mathcal{K}\}$ be a family of permutations with input-length and output-length l and k-bit key K. In [1], Bellare *et al.* define CBC mode based on the permutation family F to be the scheme $\mathrm{CBC}[F] = (\mathcal{E}\text{-CBC}, \mathcal{D}\text{-CBC}, \mathcal{K}\text{-CBC})$ with inputs that are restricted to be a multiple of the block length l. In practice, of course, plaintext messages may not satisfy this constraint, and so they need to be padded before encryption and subsequently depadded after decryption.

Let $\mathbb{B}$ denote the set $\{\{0,1\}^{il} : i \geq 0\}$ of bit strings whose length is a non-negative multiple of l, and let $\mathrm{PAD} : \{0,1\}^* \to V \subset \mathbb{B}$ and $\mathrm{DPAD} : \mathbb{B} \to \{0,1\}^* \cup \{\perp_P\}$ be functions defining a specific padding method. These mappings should both be easy to compute. We say that V is the set of all valid padded messages. For each $i \in \mathbb{N}$, let V_i be the set of messages of length il bits (i blocks) in V, so that $V = \bigcup_i V_i$. Let $I = \mathbb{B} \setminus V$ and for each $i \in \mathbb{N}$, let I_i be the set of messages of length il bits in I, so that $I = \bigcup_i I_i$. We say that I is the set of all invalid padded messages. For consistency, we require that for any $x \in I$, $\mathrm{DPAD}(x)$ returns $\perp_P$, and that $x = \mathrm{DPAD}(\mathrm{PAD}(x))$ for any $x \in \{0,1\}^*$.

Given functions $\mathrm{PAD}, \mathrm{DPAD}$ and family of permutations F, we define the scheme $\mathrm{CBC}_{PAD}[F] = (\mathcal{E}\text{-CBC}_{\mathrm{PAD}}, \mathcal{D}\text{-CBC}_{\mathrm{DPAD}}, \mathcal{K}\text{-CBC})$ as follows. This scheme uses the same key generation algorithm $\mathcal{K}$-CBC as in [1], taking as input a security parameter $k \in \mathbb{N}$ and outputting a random k-bit key K for the underlying permutation family, specifying a function F_K. We then define $\mathcal{E}\text{-CBC}_{K,\mathrm{PAD}}(x) = \mathcal{E}\text{-CBC}^{F_K}_{\mathrm{PAD}}(x)$ and $\mathcal{D}\text{-CBC}_{K,\mathrm{DPAD}}(y) = \mathcal{D}\text{-CBC}^{F_K}_{\mathrm{DPAD}}(y)$, where:

function $\mathcal{E}\text{-CBC}^{F_K}_{\mathrm{PAD}}(x)$
  $\tilde{x} = \mathrm{PAD}(x)$
  Parse $\tilde{x}$ into l-bit blocks $x_1 x_2 \ldots x_n$
  $y_0 \xleftarrow{r} \{0,1\}^l$
  for $i = 1, \ldots, n$ **do** $y_i = F_K(y_{i-1} \oplus x_i)$
  return $y_0 y_1 y_2 \ldots y_n$

function $\mathcal{D}\text{-CBC}^{F_K}_{\mathrm{DPAD}}(y)$
  Parse y into l-bit blocks as $y_0 y_1 \ldots y_n$
  for $i = 1, \ldots, n$ **do** $x_i = F_K^{-1}(y_i) \oplus y_{i-1}$
  $x = \mathrm{DPAD}(x_1 x_2 \ldots x_n)$
  return x

As an important illustrative example, we define OZ-PAD, as suggested by the ISO standard ISO/IEC 10116:2006 for use with CBC mode:

Definition 1. ***[OZ-PAD]*** *The respective padding and depadding functions are:*

function $PAD_{OZ}(x)$
  $r = |x| \mod l$
  return $\tilde{x} = x \| 1 \| 0^{l-r-1}$

function $DPAD_{OZ}(\tilde{x})$
  Parse $\tilde{x}$ *as a sequence of* l*-bit blocks* $x_1 x_2 \ldots x_n$
  if $x_n = 0^l$, ***then*** *set* $x = \perp$
  else *Parse* x_n *as* $x' \| 1 \| 0^{l-r-1}$
    Set $x = x_1 x_2 \ldots x_{n-1} x'$
  return x

Note that with this definition, a padded plaintext is invalid precisely when its last block is all-zero.

2.2 Padding Oracles

A padding oracle $\mathcal{P}(\cdot)$ indicates whether the plaintext message underlying the input ciphertext is correctly or incorrectly padded, based upon some fixed padding method and for some fixed key.

Definition 2. *A padding oracle $\mathcal{P}$, takes as input any string $y \in \{0,1\}^*$ and outputs one bit of information. If the underlying plaintext is correctly padded (i.e. lies in the set V) the oracle outputs 1; otherwise it outputs 0. Formally:*

Oracle $\mathcal{P}(y)$
  if $\mathcal{D}$-$CBC_{K,DPAD}(y) \neq \perp_P$, *then* ***return*** *1*
  else ***return*** *0*

Vaudenay [11] showed how an attacker, given repeated access to a padding oracle for CBC mode implemented with the padding method CBC-PAD, can perform decryption to recover plaintexts. His results were extended by Black and Urtubia to other padding methods in [3] and to other attack scenarios in [10,12].

2.3 Security Models

We now extend the usual security models for symmetric encryption as defined in [1] to provide the adversary with additional access to a padding oracle as in Definition 2. Here, we only consider the Chosen Plaintext Attack (CPA) setting. We generalise the existing Left-or-Right Indistinguishability, Real-or-Random Indistinguishability and Find-then-Guess Security definitions to include padding oracles, and establish the essential equivalence of these different definitions (as was done by Bellare *et al.* [1] for the usual case). Thereafter, we will prove results using the Left-or-Right Indistinguishability definition.

In fact, our models provide much stronger notions of security than are required to resist Vaudenay's original attack: roughly speaking they provide semantic security against an attacker equipped with encryption and padding oracles, whereas Vaudenay's attack only gives the attacker access to a padding oracle. This enhanced security is certainly desirable, but many padding methods used in, or recommended for, practice (e.g. OZ-PAD as recommended in ISO/IEC 10116:2006) cannot achieve our new notions. So we also introduce a weaker "one-way" notion of security to more exactly quantify the resistance of these padding methods to Vaudenay's original attack.

Note that these models could be used more generally with other modes of operation that require padding. This would be an interesting area for future work.

Left-or-Right Indistinguishability. In this model, the attacker submits two messages (x_0, x_1) to a left-or-right encryption oracle and its challenge is to determine whether it receives the encryption of x_0 or x_1 in response. In our new model, the two messages x_0, x_1 need not be of equal length but must be of equal length once they are padded, otherwise there is a trivial attack. The attacker

will also be supplied with access to a padding oracle $\mathcal{P}$ to which it may submit arbitrary strings. For $b \in \{0,1\}$, we define the left-or-right encryption oracle $\mathcal{E}\text{-CBC}_{\text{PAD},K}(\mathcal{LR}(\cdot,\cdot,b))$ by:

Oracle $\mathcal{E}\text{-CBC}_{\text{PAD},K}(\mathcal{LR}(x_0,x_1,b))$
  if $b=0$, then $C \leftarrow \mathcal{E}\text{-CBC}_{\text{PAD},K}(x_0)$
    else $(b=1)$, $C \leftarrow \mathcal{E}\text{-CBC}_{\text{PAD},K}(x_1)$
  return C

Definition 3. [LOR-PO-CPA] *Consider the encryption scheme CBC_{PAD}. Let $b \in \{0,1\}$ and $k \in \mathbb{N}$. Let $\mathcal{A}$ be an attacker that has access to the oracles $\mathcal{E}\text{-}CBC_{PAD,K}(\mathcal{LR}(\cdot,\cdot,b))$ and $\mathcal{P}(\cdot)$. The game played is as follows*

Experiment $\boldsymbol{Exp}^{lor\text{-}po\text{-}cpa-b}_{CBC_{PAD},\mathcal{A}}(k)$
  $K \xleftarrow{r} \mathcal{K}\text{-}CBC(k)$
  $b' \leftarrow \mathcal{A}^{\mathcal{E}\text{-}CBC_{PAD,K}(\mathcal{LR}(\cdot,\cdot,b)),\mathcal{P}(\cdot)}(k)$
  return b'

The attacker wins when $b'=b$, and its advantage is defined to be:

$$\boldsymbol{Adv}^{lor\text{-}po\text{-}cpa}_{CBC_{PAD},\mathcal{A}}(k) = \Pr[\boldsymbol{Exp}^{lor\text{-}po\text{-}cpa\text{-}1}_{CBC_{PAD},\mathcal{A}}(k)=1] - \Pr[\boldsymbol{Exp}^{lor\text{-}po\text{-}cpa\text{-}0}_{CBC_{PAD},\mathcal{A}}(k)=1].$$

The advantage function of the scheme is defined to be:

$$\boldsymbol{Adv}^{lor-po-cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) = \max_{\mathcal{A}}\{\boldsymbol{Adv}^{lor-po-cpa}_{CBC_{PAD},\mathcal{A}}(k)\}$$

for any integers t, q_e, μ_e, q_p. The maximum is over all adversaries $\mathcal{A}$ with time complexity t, making at most q_e queries to the encryption oracle, totalling at most $2\mu_e$ bits, and q_p queries to the padding oracle. The scheme is said to be LOR-PO-CPA secure if $\boldsymbol{Adv}^{lor-po-cpa}_{CBC_{PAD}}$ is negligible for any adversary $\mathcal{A}$ whose time complexity is polynomial in k.

Real-or-Random Indistinguishability. Here, the attacker submits a message x to a real-or-random encryption oracle and its challenge is to determine whether it receives the encryption of x or the encryption of some random r, in response. In our new model the two messages x, r need not be of equal length but must be of equal length once they are padded, otherwise a trivial attack is possible[1]. The attacker will also be supplied with access to a padding oracle $\mathcal{P}$. For $b \in \{0,1\}$, we define the real-or-random oracle $\mathcal{E}\text{-CBC}_{\text{PAD},K}(\mathcal{RR}(\cdot,b))$ by:

Oracle $\mathcal{E}\text{-CBC}_{\text{PAD},K}(\mathcal{RR}(x,b))$
  if $b=1$, then $C \leftarrow \mathcal{E}\text{-CBC}_{\text{PAD},K}(x)$
    else, $r \xleftarrow{r} \{0,1\}^*$ such that $|\mathcal{E}\text{-CBC}_K(r)| = |\mathcal{E}\text{-CBC}_K(x)|$
  $C \leftarrow \mathcal{E}\text{-CBC}_{\text{PAD},K}(r)$
  return C

[1] We must therefore assume that it is easy to generate r at random subject to this constraint; this is the case for padding methods used in practice.

Definition 4. [**ROR-PO-CPA**] *Consider the encryption scheme* CBC_{PAD}. *Let* $b \in \{0,1\}$ *and* $k \in \mathbb{N}$. *Let* $\mathcal{A}$ *be an attacker that has access to the oracles* $\mathcal{E}\text{-}CBC_{PAD,K}(\mathcal{RR}(\cdot,b))$ *and* $\mathcal{P}(\cdot)$. *The game played is as follows*

Experiment $\boldsymbol{Exp}^{ror\text{-}po\text{-}cpa-b}_{CBC_{PAD},\mathcal{A}}(k)$
$K \xleftarrow{r} \mathcal{K}\text{-}CBC(k)$
$b' \leftarrow \mathcal{A}^{\mathcal{E}\text{-}CBC_{PAD,K}(\mathcal{RR}(\cdot,b)),\mathcal{P}(\cdot)}(k)$
return b'

The attacker wins when $b' = b$ *and its advantage is defined to be:*

$$\boldsymbol{Adv}^{ror\text{-}po\text{-}cpa}_{CBC_{PAD},\mathcal{A}}(k) = \Pr[\boldsymbol{Exp}^{ror\text{-}po\text{-}cpa\text{-}1}_{CBC_{PAD},\mathcal{A}}(k) = 1] - \Pr[\boldsymbol{Exp}^{ror\text{-}po\text{-}cpa\text{-}0}_{CBC_{PAD},\mathcal{A}}(k) = 1].$$

The advantage function of the scheme is defined to be:

$$\boldsymbol{Adv}^{ror-po-cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) = \max_{\mathcal{A}}\{\boldsymbol{Adv}^{ror-po-cpa}_{CBC_{PAD},\mathcal{A}}(k)\}$$

for any integers t, q_e, μ_e, q_p. *The maximum is over all adversaries* $\mathcal{A}$ *with time complexity* t, *making at most* q_e *queries to the encryption oracle, totalling at most* μ_e *bits, and* q_p *queries to the padding oracle. The scheme is said to be ROR-PO-CPA secure if* $\boldsymbol{Adv}^{ror-po-cpa}_{CBC_{PAD}}$ *is negligible for any adversary* $\mathcal{A}$ *whose time complexity is polynomial in* k.

Find-then-Guess Security. Here, the security game is defined in two stages. First the attacker must *find* two messages x_0, x_1 upon which it wishes to be challenged. The challenger then sends y, the encryption of either x_0 or x_1, to the attacker. In the second stage, the attacker must *guess* which of these two messages x_0, x_1, is the decryption of y. Note that the two messages x_0, x_1 need not be of equal length but must be of equal length once they are padded, otherwise a trivial attack is possible.

Definition 5. [**FTG-PO-CPA**] *Consider the encryption scheme* CBC_{PAD}. *Let* $b \in \{0,1\}$ *and* $k \in \mathbb{N}$. *Let* $\mathcal{A}$ *be an attacker that has access to the oracles* $\mathcal{E}\text{-}CBC_{PAD,K}(\cdot)$ *and* $\mathcal{P}(\cdot)$. *The game played is as follows*

Experiment $\boldsymbol{Exp}^{ftg\text{-}po\text{-}cpa\text{-}b}_{CBC_{PAD},\mathcal{A}}(k)$
$K \xleftarrow{r} \mathcal{K}\text{-}CBC(k)$
$(x_0, x_1, s) \leftarrow \mathcal{A}^{\mathcal{E}\text{-}CBC_{PAD,K}(\cdot),\mathcal{P}(\cdot)}(k, find)$
$y \leftarrow \mathcal{E}\text{-}CBC_{PAD,K}(x_b)$
$b' \leftarrow \mathcal{A}^{\mathcal{E}\text{-}CBC_{PAD,K}(\cdot),\mathcal{P}(\cdot)}(k, guess, y, s)$
return b'

The attacker wins when $b' = b$ *and its advantage is defined to be:*

$$\boldsymbol{Adv}^{ftg\text{-}po\text{-}cpa}_{CBC_{PAD},\mathcal{A}}(k) = \Pr[\boldsymbol{Exp}^{ftg\text{-}po\text{-}cpa\text{-}1}_{CBC_{PAD},\mathcal{A}}(k) = 1] - \Pr[\boldsymbol{Exp}^{ftg\text{-}po\text{-}cpa\text{-}0}_{CBC_{PAD},\mathcal{A}}(k) = 1].$$

The advantage function of the scheme is defined to be:

$$\boldsymbol{Adv}^{ftg-po-cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) = \max_{\mathcal{A}}\{\boldsymbol{Adv}^{ftg-po-cpa}_{CBC_{PAD},\mathcal{A}}(k)\}$$

for any integers t, q_e, μ_e, q_p. *The maximum is over all adversaries* $\mathcal{A}$ *with time complexity* t, *making at most* q_e *queries to the encryption oracle, totalling at most* μ_e *bits, and* q_p *queries to the padding oracle. The scheme is said to be FTG-PO-CPA secure if* $\boldsymbol{Adv}^{ftg-po-cpa}_{CBC_{PAD}}$ *is negligible for any adversary* $\mathcal{A}$ *whose time complexity is polynomial in* k.

2.4 One-Way Security

As we shall see, many padding methods are not secure in the above models, but do provide a weaker form of security, in that they prevent an attacker using access to a padding oracle to perform decryption (i.e. they prevent Vaudenay's original attack). We next define a notion of security appropriate to this weaker requirement. Now the attacker's challenge is to find the decryption of a challenge ciphertext c^*. For simplicity of presentation, we focus here on the case where this ciphertext is selected uniformly at random from the set of valid ciphertexts having at most $n+1$ blocks, for some value $n = n(k)$. This uniform distribution on ciphertexts will in turn imply a padding-specific distribution $\mathcal{D}$ on the space of unpadded messages. Our definition is easily extended to the case of arbitrary distributions on this space.

Definition 6. [OW-PO-CPA] *Consider the encryption scheme* CBC_{PAD}. *Let* $k \in \mathbb{N}$. *Let* $\mathcal{A}$ *be an attacker that has access to the oracles* $\mathcal{E}$-$CBC_{PAD,K}(\cdot)$ *and* $\mathcal{P}(\cdot)$. *The game played is as follows*

Experiment $\boldsymbol{Exp}^{ow\text{-}po\text{-}cpa}_{CBC_{PAD},\mathcal{A}}(k)$
$K \xleftarrow{r} \mathcal{K}$-$CBC(k)$
$x \xleftarrow{r} \mathcal{D}$
$c^* \leftarrow \mathcal{E}$-$CBC_{PAD,K}(x)$
$x' \leftarrow \mathcal{A}^{\mathcal{E}\text{-}CBC_{PAD,K}(\cdot),\mathcal{P}(\cdot)}(k, c^*)$
return x'

The attacker wins when $x' = x$ *and its advantage is defined to be:*

$$\boldsymbol{Adv}^{ow\text{-}po\text{-}cpa}_{CBC_{PAD},\mathcal{A}}(k) = \Pr[\boldsymbol{Exp}^{ow\text{-}po\text{-}cpa}_{CBC_{PAD},\mathcal{A}}(k) = x].$$

The advantage function of the scheme is defined to be:

$$\boldsymbol{Adv}^{ow-po-cpa}_{CBC_{PAD}}(k, t, q_e, \mu_e, q_p) = \max_{\mathcal{A}}\{\boldsymbol{Adv}^{ow-po-cpa}_{CBC_{PAD},\mathcal{A}}(k)\}$$

for any integers t, q_e, μ_e, q_p. *The maximum is over all adversaries* $\mathcal{A}$ *with time complexity* t, *making at most* q_e *queries to the encryption oracle, totalling at most* μ_e *bits, and* q_p *queries to the padding oracle. The scheme is said to be OW-PO-CPA secure if* $\boldsymbol{Adv}^{ow-po-cpa}_{CBC_{PAD}}$ *is negligible for any adversary* $\mathcal{A}$ *whose time complexity is polynomial in* k.

Note that this model does allow us to study an adversary which strictly adheres to Vaudenay's original attack model, i.e. where the adversary makes no queries to the encryption oracle ($q_e = \mu_e = 0$).

2.5 Relations between Models

We can prove that the same relations between the models without padding oracles, established in [1], also hold for these new models. Proofs of the following results are given in the appendix.

Theorem 1. ***[ROR-PO-CPA ⇒ LOR-PO-CPA]*** *For the encryption scheme CBC_{PAD},*

$$\boldsymbol{Adv}^{lor\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) \leq 2.\boldsymbol{Adv}^{ror\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p).$$

Theorem 2. ***[LOR-PO-CPA ⇒ ROR-PO-CPA]*** *For the encryption scheme CBC_{PAD},*

$$\boldsymbol{Adv}^{ror\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) \leq \boldsymbol{Adv}^{lor\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p).$$

Theorem 3. ***[LOR-PO-CPA ⇒ FTG-PO-CPA]*** *For the encryption scheme CBC_{PAD},*

$$\boldsymbol{Adv}^{lor\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) \leq \boldsymbol{Adv}^{ftg\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e+1,\mu_e,q_p).$$

Theorem 4. ***[FTG-PO-CPA ⇒ LOR-PO-CPA]*** *For the encryption scheme CBC_{PAD},*

$$\boldsymbol{Adv}^{lor\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p) \leq q_e.\boldsymbol{Adv}^{ftg\text{-}po\text{-}cpa}_{CBC_{PAD}}(k,t,q_e,\mu_e,q_p).$$

This last result establishes that if a scheme has security in the Find-then-Guess sense then it is secure in the Left-or-Right sense, but the security shown is qualitatively lower.

3 Padding Methods for Chosen Plaintext Security

We are now ready to consider the security of the scheme $\mathrm{CBC}_{PAD}[F]$ for particular functions PAD, DPAD and permutation families F, in the presence of padding oracles. We recall that this scheme is already known to be insecure for many padding methods [11,3]. We divide our study of padding methods into two types:

- Padding methods with invalid paddings ($|I| > 0$);
- Padding methods with no invalid paddings ($|I| = 0$).

3.1 Padding Methods with Invalid Paddings

We recall the definition of the OZ-PAD method from Section 2.1. Here, the only invalid padded messages are those where the last block of plaintext is the all-zero block, 0^l. Since the likelihood of being able to generate a ciphertext where the last block of plaintext is 0^l is low, this suggests that OZ-PAD used with CBC mode may provide some form of security. However, an adversary can exploit this fact when performing a padding oracle attack against CBC mode using OZ-PAD in our LOR-PO-CPA attack model, as follows:

1. Choose any two distinct messages m_0, m_1 of length less than l.
2. Query the left-or-right encryption oracle on input (m_0, m_1) to obtain output of the form $y_0 y_1$.
3. Submit $(y_0 \oplus m_0)y_1$ to the padding oracle $\mathcal{P}$. If the response is 0, then return $b' = 0$, otherwise return $b' = 1$.

This very simple attack was first presented by Black and Urtubia in [3]. It shows that OZ-PAD, as recommended in ISO/IEC 10116:2006 is not secure in the sense of indistinguishability of encryptions. This attack can be applied to many other padding methods which have invalid paddings:

Lemma 1. *Consider a padding method defined by functions PAD, DPAD. Let V_1, I_1 be the sets of one-block valid and invalid messages respectively. Then the encryption scheme CBC_{PAD} is* not *LOR-PO-CPA secure if either of the following conditions is satisfied:*

1. $|V_1| > |I_1| > 0$*; or*
2. $|V_1| \geq 3$ *and* $|V_1|$, $|I_1|$ *are both odd.*

Proof. We wish to perform a similar attack as was successful against CBC mode with OZ-PAD in the LOR-PO-CPA model. In order to do this, we must find two messages m_0, m_1 such that their padded versions $p_0, p_1 \in V_1$ satisfy $p_0 \oplus m \in V_1$ and $p_1 \oplus m \in I_1$ for some mask $m \in \{0,1\}^l$: submitting (m_0, m_1) to the left-or-right encryption oracle gives an output of the form $y_0 y_1$, and then the response of the padding oracle to input $(y_0 \oplus m)y_1$ will tell us which one of m_0, m_1 was encrypted.

Proof of 1: Select p_1 arbitrarily from V_1 and any value $i_1 \in I_1$; let $m_1 = \mathrm{DPAD}(p_1)$, and set $m = p_1 \oplus i_1$, so $p_1 \oplus m = i_1 \in I_1$. The map ϕ_m defined by $\phi_m(x) = x \oplus m$ is an injective map from V_1 to $\{0,1\}^l$. So $|\phi_m(V_1)| = |V_1| > |I_1|$. This shows that, no matter the value of m, there exists some $p_0 \in V_1$ with $p_0 \oplus m \notin I_1$; hence $p_0 \oplus m \in V_1$. Now let $m_0 = \mathrm{DPAD}(p_0)$. Thus we have constructed two messages m_0, m_1 and a mask m with the required properties.

Proof of 2: Set p_0, v to be any two distinct values from V_1, and set $m = p_0 \oplus v$ and $m_0 = \mathrm{DPAD}(p_0)$. Clearly $p_0 \oplus m = v \in V_1$ and $m \neq 0^l$. We now wish to construct $p_1 \in V_1$ satisfying $p_1 \oplus m \in I_1$, and take $m_1 = \mathrm{DPAD}(p_1)$. Suppose, for a contradiction, that $p_1 \oplus m \in V_1$ for every $p_1 \in V_1$. Then, given that $m \neq 0^l$, we can divide V_1 into a covering of disjoint sets of size 2 of the form $\{p_1, p_2\}$ satisfying the equation $p_1 \oplus m = p_2$. This contradicts the fact that V_1 has odd cardinality. It follows that $p_1 \oplus m \in I_1$ for some choice of p_1 and we are done. □

Perhaps surprisingly, this result shows that any padding method having small (but non-zero) numbers of invalidly padded messages *cannot* be secure in the sense of indistinguishability. It includes the method OZ-PAD as an extremal case ($|I_1| = 1$). However, as we shall see, padding methods having no invalid padded messages at all (implying $|I_1| = 0$) are actually immune to padding oracle attacks, being secure in our LOR-PO-CPA sense.

Despite CBC mode with OZ-PAD not having LOR-PO-CPA security, we can prove that it satisfies our weaker notion of OW-PO-CPA security, assuming that the permutation family F used in its construction is a one-way permutation family (a concept which we define formally below). This implies that CBC mode with OZ-PAD provably resists Vaudenay's original attack under a rather mild assumption about the permutation family F.

Definition 7. ***[One-way Permutation Family]*** *Let $F = \{F_K : K \in \mathcal{K}\}$ be a permutation family, with input and output length l and k-bit key. Let $\mathcal{A}$ be an adversary with access to an oracle for F as defined in the experiment below:*

Experiment $\boldsymbol{Exp}^{owp}_{F,\mathcal{A}}(k)$
$K \xleftarrow{r} \mathcal{K}$
$x \xleftarrow{r} \{0,1\}^l, y \leftarrow F_K(x)$
$x' \leftarrow \mathcal{A}^{F_K(\cdot)}(y)$
return x'

The advantage of the adversary $\mathcal{A}$ is defined to be:

$$\boldsymbol{Adv}^{owp}_{F,\mathcal{A}}(k) = \Pr[\boldsymbol{Exp}^{owp}_{F,\mathcal{A}} = x].$$

The advantage function of the permutation family is defined to be:

$$\boldsymbol{Adv}^{owp}_{F}(k,t,q_F) = \max_{\mathcal{A}}\{\boldsymbol{Adv}^{owp}_{F,\mathcal{A}}(k)\}$$

where the maximum is over all $\mathcal{A}$ with time complexity t, each making at most q_F queries to the oracle for F_K.

We informally say that F is a one-way permutation family to indicate that $\mathbf{Adv}^{\mathrm{owp}}_{F}(k,t,q_F)$ is "low" for "reasonable" values of t, q_F.

Theorem 5. ***[Security of CBC_{PAD} with OZ-PAD]*** *Let $F = \{F_K : K \in \mathcal{K}\}$ be a permutation family on $\{0,1\}^l$ with a k-bit key. Let PAD and DPAD be the padding and depadding functions for OZ-PAD. Then $CBC_{PAD}[F]$ is OW-PO-CPA secure if F is a one-way permutation family. Concretely, for any t, q_e, μ_e, q_p, we have:*

$$\boldsymbol{Adv}^{ow\text{-}po\text{-}cpa}_{CBC_{PAD}[F]}(k,t,q_e,\mu_e,q_p) \leq \boldsymbol{Adv}^{owp}_{F}(k,t',q_F) + \frac{1}{2^l}.$$

where $t' = O(t)$ and $q_F \leq 1 + q_p + q_e(1 + \mu_e/l)$.

Proof. Assume we have an adversary $\mathcal{A}_1$, attacking $\mathrm{CBC}_{\mathrm{PAD}}[F]$ in the OW-PO-CPA sense. We assume that $\mathcal{A}_1$ should be given a challenge ciphertext c^* that is selected uniformly at random from the set of valid ciphertexts having at most $n+1$ blocks, for some value n. We use this adversary to construct a new adversary $\mathcal{A}_2$ that breaks the one-wayness of the permutation family F. $\mathcal{A}_2$'s input is a value $y = F_K(x)$ where K and x are selected uniformly at random, and its job is to output the value x given oracle access to F_K.

In the following construction for the adversary $\mathcal{A}_2$ we use its input y to construct a ciphertext c^* on which to challenge $\mathcal{A}_1$. $\mathcal{A}_2$ selects c^* uniformly at random from the set $\{0,1\}^{il} : 1 \leq i \leq n+1$, subject to the constraint that the last block of c^* equals y. Thus we may write $c^* = y_0 y_1 \dots y_j$ for some $j \geq 1$ and some l-bit blocks y_i, $0 \leq i \leq j$, with $y_j = y$. $\mathcal{A}_2$ then tests if $F_K(y_{j-1}) = y$ by making a query to its oracle for F_K. If this equation holds, then $\mathcal{A}_2$ has successfully inverted F_K at y, so outputs y_{j-1} and halts. Since y is uniformly distributed, this event occurs with probability at most 2^{-l}. Otherwise, we see that $y_{j-1} \oplus F_K^{-1}(y) \neq 0^l$, meaning that c^* is a valid ciphertext (because the last block of the underlying plaintext is validly padded according to the OZ-PAD method). By construction, c^* is uniformly distributed over the set of valid ciphertexts having at most $n+1$ blocks, and so can be given to $\mathcal{A}_1$ as its challenge ciphertext.

Now $\mathcal{A}_2$ needs to provide simulations of $\mathcal{A}_1$'s oracles.

To handle $\mathcal{A}_1$'s queries to the padding oracle $\mathcal{P}$ on input a $(q+1)$-block ciphertext $c = c_0 c_1 \dots c_q$, $\mathcal{A}_2$ acts as follows. $\mathcal{A}_2$ calls F_K on c_{q-1} and compares $F_K(c_{q-1})$ to c_q. If $F_K(c_{q-1}) = c_q$ and so $c_{q-1} \oplus F_K^{-1}(c_q) = 0^l$, then $\mathcal{A}_2$ outputs 0 as the response to $\mathcal{A}_1$'s query; otherwise $\mathcal{A}_2$ outputs 1. We see that $\mathcal{A}_2$'s simulation of padding oracle queries is perfect and requires one query to F_K for each query made by $\mathcal{A}_1$.

To handle $\mathcal{A}_1$'s queries to the encryption oracle on input an unpadded plaintext m, $\mathcal{A}_2$ acts as follows. First it applies the OZ-PAD padding method to m to obtain a padded plaintext $\tilde{m} = \tilde{m}_1 \tilde{m}_2 \dots \tilde{m}_q$ having q blocks for some q. Notice that q is at most $1 + |m|/l$ where $|m|$ denotes the bit-length of m. Then $\mathcal{A}_2$ selects $y_0 \xleftarrow{r} \{0,1\}^l$ and sets $y_i = F_K(y_{i-1} \oplus \tilde{m}_i)$ for each $1 \leq i \leq q$. Finally, $\mathcal{A}_2$ outputs $y_0 y_1 \dots y_q$. This is obviously a correct encryption of m for the scheme $\mathrm{CBC}_{\mathrm{PAD}}[F]$ with permutation $F_K \in F$. It is also easy to see that if $\mathcal{A}_1$ makes at most q_e encryption queries totalling at most μ_e bits, then $\mathcal{A}_2$ makes at most $q_e(1 + \mu_e/l)$ queries to its oracle for F_K in handling these encryption queries.

The final output of $\mathcal{A}_1$ is a guess at the plaintext corresponding to the decryption of c^*. Let b denote the last block of the OZ-PAD padded version of $\mathcal{A}_1$'s output. If $\mathcal{A}_1$'s output is correct, then we have the decryption equation $b = y_{j-1} \oplus F_K^{-1}(y)$. From this equation we have $F_K^{-1}(y) = b \oplus y_{j-1}$ and so $\mathcal{A}_2$ outputs $b \oplus y_{j-1}$ as its guess for x. We see that $\mathcal{A}_2$ is correct if $\mathcal{A}_1$ is. Hence $\mathcal{A}_2$'s success probability is the same as $\mathcal{A}_1$'s.

Counting the total number of oracle queries made by $\mathcal{A}_2$ and estimating its running time as $O(t)$ completes the proof. □

From the above theorem, we can say that CBC mode used with OZ-PAD does offer security against Vaudenay's original attack. Despite this, and as we previously showed, OZ-PAD does not offer security in the stronger indistinguishability sense that we desire. We therefore recommend using a padding method which does offer this greater level of security. We now turn our attention to such methods.

3.2 Padding Methods With No Invalid Paddings

We first provide some examples of padding methods which have no invalid padded messages (i.e. $|I| = 0$).

Black and Urtubia [3] suggest the use of the Arbitrary Tail padding method. Below, we define both the bit- and byte-oriented versions of this padding method; the latter only applies for messages that are guaranteed to be a whole number of bytes, and so does not strictly comply with our bit-oriented definitions.

Definition 8. ***[Abit Pad]*** *Study the last bit b of the message m. Pad the message with the opposite of bit b. At least one padding bit must be added. For the empty string $m = \epsilon$, pad with either* 0 *or* 1 *with equal probability.*

Note that we slightly alter Black and Urtubia's original definition of Abit Pad to pad the empty string with a randomly chosen bit 0 or 1. This eliminates the possibility of 1^l being an invalid padded message. With this modification, Abit Pad has no invalid padded messages.

Definition 9. ***[Abyte Pad]*** *For a message m with k complete bytes, study the last byte X of the message m. Pick some distinct random byte Y and pad the message with this byte. At least one padding byte must be added.*

Again, Abyte Pad has no invalid padded messages.

We prove the following simple result concerning the security of all padding methods which have no invalid padded messages.

Theorem 6. ***[Security of CBC_{PAD} for a padding method containing no invalid paddings]*** *Suppose F is a permutation family with length l. Let $PAD, DPAD$ be the padding and depadding functions for some padding method with no invalid paddings. If CBC is LOR-CPA secure then CBC_{PAD} is LOR-PO-CPA secure. Concretely, for any t, q_e, q_p, μ_e*

$$\boldsymbol{Adv}^{lor\text{-}po\text{-}cpa}_{CBC_{PAD}[F]}(k, t, q_e, \mu_e, q_p) \leq \boldsymbol{Adv}^{lor\text{-}cpa}_{CBC[F]}(k, t, q_e, \mu_e).$$

Proof. Assume we have an adversary $\mathcal{A}_1$ that attacks $\mathrm{CBC_{PAD}}$ in the LOR-PO-CPA sense. We then use this adversary to construct a new adversary $\mathcal{A}_2$ to attack CBC in the LOR-CPA sense.

$\mathcal{A}_2$ will use its encryption oracle to provide a simulation of $\mathcal{A}_1$'s encryption oracle on input x. $\mathcal{A}_2$ first runs PAD on x, and sends the padded message to its encryption oracle, so $\mathcal{E}\text{-}\mathrm{CBC}_{K,\mathrm{PAD}}(x) = \mathcal{E}\text{-}\mathrm{CBC}_K(\mathrm{PAD}(x))$. Since the padding method being used has no invalid padded messages this means that any chosen ciphertext corresponds to a valid padded plaintext and so $\mathcal{A}_2$ models $\mathcal{A}_1$'s padding oracle by returning 1 to any query. Finally, $\mathcal{A}_2$ outputs whatever $\mathcal{A}_1$ outputs. It is evident that $\mathcal{A}_2$'s success probability is equal to that of $\mathcal{A}_1$ and the result follows. □

Note that this theorem can be strengthened to show that

$$\mathbf{Adv}^{\text{lor-po-cpa}}_{\mathrm{CBC_{PAD}}[F]}(k, t, q_e, \mu_e, q_p) = \mathbf{Adv}^{\text{lor-cpa}}_{\mathrm{CBC}[F]}(k, t, q_e, \mu_e)$$

under the assumption that the padding method satisfies $\mathrm{PAD}(\mathrm{DPAD}(y)) = y$ for all y. Any deterministic padding scheme meets this requirement.

From the results of Bellare *et al.* [1], we know that CBC mode *without* padding is secure in the LOR-CPA sense if F is a pseudo-random permutation family. From the above theorem, we see that $\mathrm{CBC}_{PAD}[F]$ is secure in the LOR-PO-CPA sense under the same condition on F, for any padding method having $|I| = 0$. Given their strong security guarantees, we therefore recommend the use of the methods such as the Abit Pad method with CBC mode.

4 Conclusion

In this paper we have extended existing security models for CBC mode encryption to incorporate padding and attacks based on padding oracles in the chosen plaintext setting. We have then used these models to study the security of particular padding methods. We proved that, for a large number of padding methods where invalid padded messages do exist, a trivial attack can be performed against CBC mode using such a method. We showed that the OZ-PAD method recommended for use with CBC mode in ISO/IEC 10116:2006 provably resists Vaudenay's original padding oracle attack, even though it does not attain our strongest indistinguishability notion. We also showed that *any* padding method that has no invalid padded messages automatically achieves immunity against padding oracle attacks, as it meets our IND-PO-CPA notion when combined with CBC mode encryption built using a pseudo-random permutation family. The Abit pad method is a very simple padding method with this property.

Given the results of this paper, we suggest that any future version of the ISO/IEC 10116 standard be revised to recommend the use of a padding method such as Abit pad in place of OZ-PAD.

In a companion paper to this, we further develop the theory of padding oracle attacks and security models to cover the case of CCA security. We also examine the question of how encryption and padding should be combined with integrity protection in order to provably defeat padding oracle attacks, even in the face of imperfect implementations of the type studied in [5].

References

1. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 1997), pp. 394–403. IEEE, Los Alamitos (1997)
2. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-MAC paradigm. ACM Transactions on Information and Systems Security 7, 206–241 (2004)
3. Black, J., Urtubia, H.: Side-channel attacks on symmetric encryption schemes: The case for authenticated encryption. In: Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002, pp. 327–338 (2002)
4. Boldyreva, A., Kumar, V.: Provable-security analysis of authenticated encryption in kerberos. In: IEEE Symposium on Security and Privacy, pp. 92–100. IEEE Computer Society, Los Alamitos (2007)

5. Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS channel. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 583–599. Springer, Heidelberg (2003)
6. Degabriele, J.P., Paterson, K.G.: Attacking the IPsec standards in encryption-only configurations. In: IEEE Symposium on Security and Privacy, pp. 335–349. IEEE Computer Society, Los Alamitos (2007)
7. Kent, S.: IP encapsulating security payload (ESP). RFC 4303 (December 2005)
8. Krawczyk, H.: The order of encryption and authentication for protecting communications (or: How secure is SSL?). In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 310–331. Springer, Heidelberg (2001)
9. Mitchell, C.J.: Error oracle attacks on CBC mode: Is there a future for CBC mode encryption? In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 244–258. Springer, Heidelberg (2005)
10. Paterson, K.G., Yau, A.K.L.: Padding oracle attacks on the ISO CBC mode encryption standard. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 305–323. Springer, Heidelberg (2004)
11. Vaudenay, S.: Security flaws induced by CBC padding – applications to SSL, IPSEC, WTLS. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 534–546. Springer, Heidelberg (2002)
12. Yau, A.K.L., Paterson, K.G., Mitchell, C.J.: Padding oracle attacks on CBC-mode encryption with secret and random IVs. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 299–319. Springer, Heidelberg (2005)
13. Zhou, Y., Feng, D.: Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. Cryptology ePrint Archive, Report 2005/388 (2005), `http://eprint.iacr.org/`

Appendix

Proof (of Theorem 1). Assume $\mathcal{A}_1$ is an adversary attacking $\mathrm{CBC}_{\mathrm{PAD}}$ in the LOR-PO-CPA sense. We can use this adversary to construct a new adversary $\mathcal{A}_2$ attacking $\mathrm{CBC}_{\mathrm{PAD}}$ in the ROR-PO-CPA sense.

Let $\mathcal{E}_2(\cdot)$ be $\mathcal{A}_2$'s encryption oracle and $\mathcal{P}(\cdot)$ its padding oracle. $\mathcal{A}_2$ will run $\mathcal{A}_1$ using its oracles to provide a simulation of $\mathcal{A}_1$'s oracles.

For $b \in \{0,1\}$ and any x_0, x_1, we define $\mathcal{E}_1(\mathcal{LR}(x_0, x_1, b))$ to be $\mathcal{E}_2(x_b)$. So when $\mathcal{A}_1$ queries its encryption oracle with $\mathcal{E}_1(\mathcal{LR}(x_0, x_1, b))$ then $\mathcal{A}_2$ will respond with the response given by the output given by its oracle for $\mathcal{E}_2(x_b)$.

Algorithm $\mathcal{A}_2^{\mathcal{E}_2(\cdot),\mathcal{P}(\cdot)}(k)$
  let $b \xleftarrow{r} \{0,1\}$
  if $b = 0$ then $d \leftarrow \mathcal{A}_1^{\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,0)),\mathcal{P}(\cdot)}(k)$
    else $d \leftarrow \mathcal{A}_1^{\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,1)),\mathcal{P}(\cdot)}(k)$
  if $b = d$ **return** 1
    else **return** 0

Now we find A_2's advantage. We have:

$$\mathbf{Adv}^{\text{ror-po-cpa}}_{\mathrm{CBC}_{\mathrm{PAD}},A_2}(k) = \Pr[\mathbf{Exp}^{\text{ror-po-cpa-1}}_{\mathrm{CBC}_{\mathrm{PAD}},A_2}(k) = 1] - \Pr[\mathbf{Exp}^{\text{ror-po-cpa-0}}_{\mathrm{CBC}_{\mathrm{PAD}},A_2}(k) = 1]$$

Let us first consider $\Pr[\mathbf{Exp}^{\text{ror-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k) = 1]$. In this case the encryption oracle $\mathcal{E}_2(\cdot) = \mathcal{E}_2(\mathcal{RR}(x_b, 0))$ returns the encryption of a randomly chosen plaintext r. Therefore since we defined $\mathcal{E}_1(\mathcal{LR}(x_0, x_1, b))$ to be $\mathcal{E}_2(x_b)$, this means that $\mathcal{E}_1(\mathcal{LR}(x_0, x_1, 0))$ and $\mathcal{E}_1(\mathcal{LR}(x_0, x_1, 1))$ return identically distributed answers for any (x_0, x_1). The encryption oracle for this case therefore only outputs ciphertexts of randomly chosen plaintexts. This means that the padding oracle can only be called on a randomly generated ciphertext or on the random output from the encryption oracle. The Padding Oracle therefore only provides information relating to the random ciphertext and its underlying plaintext and does not output any information which can be used to distinguish between the plaintext inputs x_0, x_1. We can therefore say that any input from the Padding Oracle is independent of the bit b. Thus the Padding Oracle gives no assistance in determining x_b. Therefore $\Pr[\mathbf{Exp}^{\text{ror-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k) = 1] = \frac{1}{2}$. Hence,

$$
\begin{aligned}
&\mathbf{Adv}^{\text{ror-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2} \\
&= \Pr[\mathbf{Exp}^{\text{ror-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k) = 1] - \tfrac{1}{2} \\
&= \tfrac{1}{2}\Pr[\mathbf{Exp}^{\text{lor-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1] - \tfrac{1}{2}\Pr[\mathbf{Exp}^{\text{lor-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 0] - \tfrac{1}{2} \\
&= \tfrac{1}{2}\Pr[\mathbf{Exp}^{\text{lor-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1] - \tfrac{1}{2}(1 - \Pr[\mathbf{Exp}^{\text{lor-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1]) - \tfrac{1}{2} \\
&= \tfrac{1}{2}(\Pr[\mathbf{Exp}^{\text{lor-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{lor-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1]) \\
&= \tfrac{1}{2}\mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}
\end{aligned}
$$

Since $\mathbf{Exp}^{\text{ror-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k) = 1$ provides a perfect simulation of $\mathcal{A}_1$ against the LOR-PO-CPA game. The algorithm simulating $\mathcal{A}_2$ outputs 1 only when $b = d$ and this is the same as $\mathcal{A}_1$ winning the LOR-PO-CPA game.

Since $\mathcal{A}_1$ is an arbitrary adversary we have,

$$\mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}}}(k, t, q_e, \mu_e, q_p) \leq 2.\mathbf{Adv}^{\text{ror-po-cpa}}_{\text{CBC}_{\text{PAD}}}(k, t, q_e, \mu_e, q_p). \qquad \square$$

Proof (of Theorem 2). Assume $\mathcal{A}_2$ is an adversary attacking CBC_{PAD} in the ROR-PO-CPA sense. We can use this adversary to construct a new adversary $\mathcal{A}_1$ attacking CBC_{PAD} in the LOR-PO-CPA sense.

Let $\mathcal{E}_1(\cdot)$ be $\mathcal{A}_1$'s encryption oracle and $\mathcal{P}(\cdot)$ its padding oracle. $\mathcal{A}_1$ will run $\mathcal{A}_2$ using its oracles to provide a simulation of $\mathcal{A}_2$'s oracles.

For any $x \in \{0,1\}^*$, define $\mathcal{E}_2(x)$ to be $\mathcal{E}_1(r, x)$, where r is chosen randomly for each call to the oracle. So when $\mathcal{A}_2$ queries its encryption oracle with $\mathcal{E}_2(x)$ then $\mathcal{A}_1$ will respond with the response given by the output given by its oracle for $\mathcal{E}_1(r, x)$.

Algorithm $\mathcal{A}_1^{\mathcal{E}_1(\cdot,\cdot),\mathcal{P}(\cdot)}(k)$
  return $\mathcal{A}_2^{\mathcal{E}_2(\cdot),\mathcal{P}(\cdot)}(k)$

If $\mathcal{A}_2$'s hidden bit b is 0, then all responses from the encryption oracle are random and this corresponds to the left response in $\mathcal{A}_1$'s encryption oracle. Similarly, if $\mathcal{A}_2$'s hidden bit b is 1, then all responses are real and this corresponds to the right response in $\mathcal{A}_1$'s encryption oracle. So $\mathcal{A}_1$'s advantage will be:

$$\begin{aligned}\mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) &= \Pr[\mathbf{Exp}^{\text{lor-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)=1] - \Pr[\mathbf{Exp}^{\text{lor-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)=1] \\ &= \Pr[\mathbf{Exp}^{\text{ror-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k)=1] - \Pr[\mathbf{Exp}^{\text{ror-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k)=1] \\ &= \mathbf{Adv}^{\text{ror-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_2}(k)\end{aligned}$$

Since $\mathcal{A}_2$ is an arbitrary adversary we have,

$$\mathbf{Adv}^{\text{ror-po-cpa}}_{\text{CBC}_{\text{PAD}}}(k,t,q_e,\mu_e,q_p) \leq \mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}}}(k,t,q_e,\mu_e,q_p). \qquad \square$$

Proof (of Theorem 3). Assume $\mathcal{A}_3$ is an adversary attacking CBC_{PAD} in the FTG-PO-CPA sense. We can use this adversary to construct a new adversary $\mathcal{A}_1$ attacking CBC_{PAD} in the LOR-PO-CPA sense.

Let $\mathcal{E}_1(\cdot)$ be $\mathcal{A}_1$'s encryption oracle and $\mathcal{P}(\cdot)$ its padding oracle. $\mathcal{A}_1$ will run $\mathcal{A}_3$ using its oracles to provide a simulation of $\mathcal{A}_3$'s oracles.

For any $x \in \{0,1\}^*$, define $\mathcal{E}_3(x)$ to be $\mathcal{E}_1(x,x)$. So when $\mathcal{A}_3$ queries its encryption oracle with $\mathcal{E}_3(x)$ then $\mathcal{A}_1$ will respond with the response given by the output given by its oracle for $\mathcal{E}_1(x,x)$.

Algorithm $\mathcal{A}_1^{\mathcal{E}_1(\cdot,\cdot),\mathcal{P}(\cdot)}(k)$
  Let $(x_0,x_1,s) \leftarrow \mathcal{A}_3^{\mathcal{E}_3(\cdot,\cdot),\mathcal{P}(\cdot)}(k,\text{find})$
  return $\mathcal{A}_3^{\mathcal{E}_3(\cdot),\mathcal{P}(\cdot)}(k,\text{guess},\mathcal{E}_1(x_0,x_1),s)$

If $\mathcal{A}_3$'s hidden bit b is 0, then all responses from the encryption oracle correspond to the left response in $\mathcal{A}_1$'s encryption oracle. Similarly, if $\mathcal{A}_2$'s hidden bit b is 1, then all responses correspond to the right response in $\mathcal{A}_1$'s encryption oracle. So $\mathcal{A}_1$'s advantage will be:

$$\begin{aligned}\mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) &= \Pr[\mathbf{Exp}^{\text{lor-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)=1] - \Pr[\mathbf{Exp}^{\text{lor-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)=1] \\ &= \Pr[\mathbf{Exp}^{\text{ftg-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_3}(k)=1] - \Pr[\mathbf{Exp}^{\text{ftg-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_3}(k)=1] \\ &= \mathbf{Adv}^{\text{ftg-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_3}(k)\end{aligned}$$

Since $\mathcal{A}_3$ is an arbitrary adversary we have:

$$\mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}}}(k,t,q_e,\mu_e,q_p) \leq \mathbf{Adv}^{\text{ftg-po-cpa}}_{\text{CBC}_{\text{PAD}}}(k,t,q_e+1,\mu_e,q_p). \qquad \square$$

Proof (of Theorem 4). Assume $\mathcal{A}_1$ is an adversary attacking CBC_{PAD} in the LOR-PO-CPA sense. We can use this adversary to construct a new adversary $\mathcal{A}_3$ that attacks CBC_{PAD} in the FTG-PO-CPA sense.

Let $\mathcal{E}_3(\cdot)$ be $\mathcal{A}_3$'s encryption oracle and $\mathcal{P}(\cdot)$ its padding oracle. $\mathcal{A}_3$ will run $\mathcal{A}_1$ using its oracles to provide a simulation of $\mathcal{A}_1$'s oracles.

For $b \in \{0,1\}$ and any x_0, x_1, we define $\mathcal{E}_1(\mathcal{LR}(x_0,x_1,b))$ to be $\mathcal{E}_3(x_b)$. So when $\mathcal{A}_1$ queries its encryption oracle with $\mathcal{E}_1(\mathcal{LR}(x_0,x_1,b))$ then $\mathcal{A}_3$ will respond with the output given by its oracle for $\mathcal{E}_3(x_b)$.

Algorithm $\mathcal{A}_3^{\mathcal{E}_3(\cdot),\mathcal{P}(\cdot)}(k,\text{find})$

- Let $i \xleftarrow{r} \{1,\ldots,q_e\}$
- Run $\mathcal{A}_1$ answering its encryption oracle queries with $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,0)) = \mathcal{E}_3(x_0)$ and any padding oracle queries with $\mathcal{P}(\cdot)$, until the i-th encryption oracle query, which we denote (x_0^i, x_1^i). (Where $\mathcal{A}_1$ has made the query and is waiting for the oracle to respond.) $\mathcal{A}_1$'s state at this point is denoted by s.
- **return** (x_0^i, x_1^i, s)

Now $\mathcal{A}_3$'s challenger selects a bit $b \xleftarrow{r} \{0,1\}$ and gives $\mathcal{A}_3$ the value y, where $y = \mathcal{E}_3(x_b)$.

Algorithm $\mathcal{A}_3^{\mathcal{E}_3(\cdot),\mathcal{P}(\cdot)}(k, \text{guess}, y, s)$

- Resume execution of $\mathcal{A}_1$ in state s by answering the i-th encryption query (x_0^i, x_1^i) with y then stop before another oracle query is made.
  (*Note:* $y = \mathcal{E}_1(\mathcal{LR}(x_0^i, x_1^i, b)) = \mathcal{E}_3(x_b^i)$)
- Continue execution of $\mathcal{A}_1$, answering all encryption oracle queries now with $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,1)) = \mathcal{E}_3(x_1)$ and any padding oracle queries with $\mathcal{P}(\cdot)$, until A_1 halts.
- **if** $\mathcal{A}_1$ outputs 1 then **return** 1, else **return** 0

Now define a sequence of $q_e + 1$ experiments, for $j = 0, \ldots, q_e$:

Experiment $\mathbf{Exp}^{\text{hyb-atk-}j}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)$
  $K \xleftarrow{r} \mathcal{K}(k)$
  Run $\mathcal{A}_1$
  Answer first j encryption oracle queries of $\mathcal{A}_1$ by $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,0))$
  Answer the remaining encryption oracle queries by $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,1))$
  Answer any padding oracle queries by $\mathcal{P}(\cdot)$
  return output of $\mathcal{A}_1$

We can now think of the experiment $\mathbf{Exp}^{\text{ftg-po-cpa-}b}_{\text{CBC}_{\text{PAD}},\mathcal{A}_3}(k)$ in terms of this new experiment $\mathbf{Exp}^{\text{hyb-atk-}j}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)$. If $b = 0$ then $y = \text{CBC}_K(x_0^i)$ and so the first $i+1$ responses from $\mathcal{A}_1$'s encryption oracle are $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,0))$, while the remaining responses are $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,1))$. Therefore $\mathcal{A}_1$'s output will be the same as the output of $\mathbf{Exp}^{\text{hyb-atk-}i+1}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)$.

On the other hand if $b = 1$ then $y = \text{CBC}_K(x_1^i)$ and so the first i responses from $\mathcal{A}_1$'s encryption oracle are $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,0))$, while the remaining responses are $\mathcal{E}_1(\mathcal{LR}(\cdot,\cdot,1))$. Therefore $\mathcal{A}_1$'s output will be the same as the output of $\mathbf{Exp}^{\text{hyb-atk-}i}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)$.

Note that there are two special cases of this new experiment $\mathbf{Exp}^{\text{hyb-atk-}j}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)$. If $j = 0$ then the experiment is the same as an LOR-PO-CPA game when the random bit b is 0. Similarly, if $j = q_e$ then the experiment is the same as an LOR-PO-CPA game when the random bit b is 1.

Since i is chosen at random from $\{1, \ldots, q_e\}$, $\mathcal{A}_3$'s advantage is determined as the average of all the advantages over the random choice of i. Hence:

$$
\begin{aligned}
&\mathbf{Adv}^{\text{ftg-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_3}(k) \\
&= \tfrac{1}{q_e} \cdot \textstyle\sum_{i=0}^{q_e-1} (\Pr[\mathbf{Exp}^{\text{hyb-atk-}i}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{hyb-atk-}i+1}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1]) \\
&= \tfrac{1}{q_e} \cdot (\Pr[\mathbf{Exp}^{\text{hyb-atk-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{hyb-atk-}q_e}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1]) \\
&= \tfrac{1}{q_e} \cdot (\Pr[\mathbf{Exp}^{\text{lor-po-cpa-1}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1] - \Pr[\mathbf{Exp}^{\text{lor-po-cpa-0}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k) = 1]) \\
&= \tfrac{1}{q_e} \cdot \mathbf{Adv}^{\text{lor-po-cpa}}_{\text{CBC}_{\text{PAD}},\mathcal{A}_1}(k)
\end{aligned}
$$

Since $\mathcal{A}_1$ is an arbitrary adversary, the claimed relation holds. □

Constructing Strong KEM from Weak KEM (or How to Revive the KEM/DEM Framework)

Joonsang Baek[1], David Galindo[2], Willy Susilo[3], and Jianying Zhou[1]

[1] Cryptography and Security Department
Institute for Infocomm Research, Singapore
{jsbaek,jyzhou}@i2r.a-star.edu.sg
[2] Computer Science Department, University of Malaga
dgalindo@lcc.uma.es
[3] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia
wsusilo@uow.edu.au

Abstract. We propose a generic method that transforms a weakly secure KEM, i.e. a KEM which is secure against constrained chosen ciphertext attack (CCCA), to a strongly secure KEM, i.e. a KEM which is secure against full chosen ciphertext attack (CCA). The proposed method does not depend on the random oracle nor any other non-standard assumptions. Using this method, we obtain new efficient hybrid encryption schemes based on Kurosawa&Desmedt and Hofheinz&Kiltz weakly secure KEMs. These are the first hybrid encryption schemes which are as efficient as Kurosawa&Desmedt and Hofheinz&Kiltz encryption schemes, but whose security can be explained in the original KEM/DEM framework.

1 Introduction

1.1 Motivation

As a systematic approach to build hybrid public key encryption schemes[1], the KEM/DEM (Key Encapsulation Mechanism/Data Encryption Mechanism) framework was introduced by Cramer and Shoup [9]. The KEM/DEM framework captures the most basic intuition about hybrid encryption schemes: the KEM on input a public key as input generates a DEM-key and the DEM encrypts a plaintext message using the DEM-key generated by the KEM. In terms of security, [9] shows that if the KEM is secure against chosen ciphertext attack (CCA-secure) and DEM is one-time CCA-secure, the resulting hybrid encryption scheme is CCA-secure[2].

[1] Hereinafter, we simply call them "hybrid encryption schemes".

[2] The term CCA-secure is somewhat abused here, since it has different (but related) meanings depending on whether it refers to public key encryption, KEM or DEM schemes.

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 358–374, 2008.

However, it turned out the CCA-security for both KEM and DEM is not a *necessary* condition but a mere sufficient one. One of the famous examples is Kurosawa and Desmedt's [17] hybrid encryption scheme: Although the KEM part of Kurosawa and Desmedt's hybrid encryption scheme is *not* CCA-secure as shown in [14], the hybrid encryption scheme itself is proven to be CCA-secure [17,13]. It seemed that the KEM/DEM framework had to be modified or extended to capture a larger class of hybrid encryption schemes.

Indeed, Abe et al. [1,2] introduced another framework, called "Tag-KEM/DEM". Different from the normal KEM, the Tag-KEM takes arbitrary string called "tag" as input. In the Tag-KEM/DEM framework, the DEM part becomes a tag. It was shown in [2] that the CCA-security of Kurosawa and Desmedt's hybrid encryption scheme can fully be explained in the Tag-KEM/DEM framework.

Recently, Hofheinz and Kiltz [15] proposed another framework which combines a KEM secure against "constrained chosen ciphertext attack (CCCA)", which is weaker than usual CCA on KEM[3], with a DEM which is one-time secure authenticated encryption (AE-OT) to yield a CCA-secure hybrid encryption scheme.

One advantage of the KEM/DEM framework, as noted in [11], is that the security requirements of the asymmetric and symmetric parts of the hybrid encryption scheme can be completely separated and studied independently. Additionally, as pointed out in [22] and [2], the KEM/DEM framework is suitable for *streaming* applications since the receiver does not need to buffer the entire ciphertext. Indeed, the KEM/DEM approach to construct hybrid encryption schemes has gained popularity among researchers and has been supported by several standards including the ISO/IEC standard on public key encryption [16].

We argue that in spite of the new frameworks' effectiveness in explaining some hybrid encryption schemes, the original KEM/DEM framework is still very attractive as it is standardized, it allows "stream processing" and gives the users flexibility in choosing the symmetric encryption algorithm for DEM. On the contrary, the Tag-KEM/DEM framework does not provide stream processing *in general* and the new framework proposed in [15] requires DEM to be one-time AE-OT, which is strictly stronger than one-time CCA-secure symmetric encryption.

Last but not least, there is in our opinion a pedagogical reason for sticking to the original KEM/DEM framework. The KEM/DEM framework is intuitive and easier to understand by the general information security public, in particular by implementors. We think there is no need for individuals outside the cryptographic community to update their cryptographic knowledge at every new theoretical advancement. Providing intuitive sound and stable cryptographic frameworks could help to obtain less error-prone end-systems.

1.2 Our Contributions

First, we present a simple but useful method that transforms a CCCA-secure KEM to a CCA-secure KEM. The basic idea behind this transformation is to

[3] Readers are referred to Section 2.1 for the detailed reviews on the CCA and CCCA notions of KEM.

authenticate the ciphertext output by the CCCA-secure KEM using a secure Message Authentication Code (MAC) with a (symmetric) key generated by the CCCA-secure KEM.

Since the KEM parts of both Kurosawa and Desmedt's [17], and Hofheinz and Kiltz's [15] hybrid encryption schemes are known to be CCCA-secure [15], our transformation yields two new CCA-secure KEMs based the Decisional Diffie Hellman (DDH) problem. We discuss in Section 4.2 that these KEM schemes are the most efficient *CCA-secure* KEMs in the literature to our knowledge.

We also obtain new hybrid encryption schemes by combining our new KEM schemes with *any* one-time CCA-secure DEMs. However, a particular interesting case is when they are combined with length-preserving CCA-secure DEMs [18,21]. In this case, the resulting hybrid encryption schemes are as efficient as Kurosawa-Desmedt's [13][4] and Hofheinz and Kiltz's [15] hybrid encryption schemes in terms of ciphertext length and computation time. In contrast to the latter, the CCA-security of the new schemes can be explained in the original KEM/DEM framework.

1.3 Further Discussion on Related Work

The concept of chosen ciphertext security for public key encryption was first introduced by Naor and Yung [19] and further developed by Dolev, Dwork and Naor [12]. Relations between the non-malleability and indistinguishability frameworks for chosen ciphertext security were clarified by Bellare et al. [4]. As mentioned previously, the KEM/DEM framework for constructing hybrid encryption was introduced by Cramer and Shoup [9].

After the KEM/DEM framework emerged, Abe et al. [1] proposed a new framework, called "Tag-KEM/DEM". Different from KEMs, the encapsulation algorithm [1,2] of a Tag-KEM takes an arbitrary string called "tag" as input Subsequently, a hybrid encryption scheme can be built regarding a DEM ciphertext as a tag [1].

More recently, Hofheinz and Kiltz [15] introduced a new security notion for KEM, "security against constrained chosen ciphertext attacks (CCCA)". This notion is *weaker* than the normal chosen ciphertext security for KEM. In CCCA an attacker makes a decapsulation query by presenting a ciphertext together with a boolean predicate, which represents some a-priori knowledge about the decapsulated key. Hofheinz and Kiltz showed that a hybrid encryption scheme in which a DEM encrypts a plaintext message using a key output by the CCCA-secure KEM is CCA-secure if the underlying DEM is a one-time secure authenticated encryption scheme (AE-OT secure). However, as pointed out in Appendix D of

[4] The original construction of Kurosawa and Desmedt's hybrid encryption scheme given in [17] is based on the information theoretically secure key derivation function (KDF) and message authentication code (MAC). In [13], it is shown that these primitives can be replaced by computationally secure ones. Throughout this paper, "Kurosawa and Desmedt's hybrid encryption scheme" refers to the scheme with computationally secure KDF and MAC discussed in [13].

[15], authenticated encryption is a strictly *stronger* security notion than chosen-ciphertext security for symmetric encryption as shown in [5]. Moreover, length-preserving authenticated encryption does not exist while length-preserving CCA-secure symmetric encryption does.

As a variant of Kurosawa and Desmedt's KEM scheme, Okamoto [20] proposed a new KEM scheme. Although this KEM scheme is validity-check-free, it requires a special type of pseudo random function, called "πPRF". However constructing a πPRF family is still an open problem [20].

2 Preliminaries

In this section, we review the primitives used throughout this paper.

First, we introduce some notations. We use the notation $\mathsf{A}(\cdot, \ldots, \cdot)$ to denote an algorithm, with input arguments separated by commas. (Note that our underlying computational model is a probabilistic Turing Machine.) We use $a \leftarrow \mathsf{A}(x_1, \ldots, x_n)$ to denote the assignment of an element from the output of A on input $(x_1, \ldots, x_n)$ to the variable a. By "$\stackrel{\text{par}}{\rightarrow}$" we denote "parsing operation". For instance, $a \stackrel{\text{par}}{\rightarrow} (b, c)$ means that a string a is parsed as (b, c).

2.1 Key Encapsulation Mechanism (KEM) and Its Security Notions

Key Encapsulation Mechanism (KEM). The KEM scheme, denoted KEM, consists of the following algorithms [9,22,16].

- $\mathsf{KEM.Gen}(1^\lambda)$: A probabilistic algorithm that generates a public/private key pair (pk, sk).
- $\mathsf{KEM.Encap}(pk)$: A probabilistic algorithm that generates a ciphertext/key pair (ϕ, K).
- $\mathsf{KEM.Decap}(sk, \phi)$: An algorithm that outputs either a key K or a special symbol $\perp$ (meaning "reject").

CCA-Security for KEM. The security notion for KEM against (adaptive) chosen ciphertext attack, is defined as follows. Let $\mathcal{A}$ be an attacker. Consider the following game in which $\mathcal{A}$ interacts with the challenger.

> **Phase 1:** The challenger runs the key generation algorithm providing 1^λ as input to generate a public/private key pair (pk, sk). The challenger then computes a challenge ciphertext ϕ^* and a key K_1^* by running the encapsulation algorithm. It also picks $K_0^* \in S_K$ at random, where S_K denotes the key space. It then picks $\beta \in \{0, 1\}$ at random and gives (pk, ϕ^*, K_β^*) to $\mathcal{A}$.
> **Phase 2:** A submits ciphertexts, each of which is denoted by ϕ. On receiving ϕ, the challenger runs the decapsulation algorithm on input ϕ and passes the resulting decapsulation to $\mathcal{A}$. At the end of this phase, $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$.

We define the attacker $\mathcal{A}$'s success probability by

$$\mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{A},\mathsf{KEM}}(\lambda) = \Big| \Pr[\beta' = \beta] - \frac{1}{2} \Big|.$$

We say that KEM is CCA-secure if $\mathbf{Adv}^{\mathrm{CCA}}_{\mathsf{KEM}}(\lambda) = \max_{\mathcal{A}} \left\{ \mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{A},\mathsf{KEM}}(\lambda) \right\}$ is negligible for any attacker $\mathcal{A}$.

Constrained CCA-Security for KEM. Hofheinz and Kiltz [15] defined a new security notion for KEM, called constrained chosen-ciphertext security (CCCA). This is a relaxed security notion in which an attacker is allowed to make a decapsulation query only if it has some a-priori knowledge about the decapsulated key. This is modeled by letting the attacker specify an efficiently computable boolean predicate $\mathrm{pred} : S_K \to \{0,1\}$, where S_K denotes the key space of key K embedded in a given ciphertext ϕ. Then, the decapsulated key K is returned only if $\mathrm{pred}(K) = 1$; otherwise $\perp$ is returned.

Formally, let $\mathcal{A}$ be the attacker. On receiving the decapsulation query consisting on a pair of ciphertext and predicate (ϕ, pred), the "constrained decapsulation oracle" CDecap outputs $K = \mathsf{KEM.Decap}(sk, \phi)$ if $\mathrm{pred}(K) = 1$ and otherwise it outputs $\perp$. Now consider the following game.

Phase 1: The challenger runs the key generation algorithm providing 1^λ as input to generate a public/private key pair (pk, sk). The challenger then computes a challenge ciphertext ϕ^* and a key K_1^* by running the encapsulation algorithm. It also picks $K_0^* \in S_K$ at random. It then picks $\beta \in \{0,1\}$ at random and gives (pk, ϕ^*, K_β^*) to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ submits a pair of ciphertext and predicated, each of which is denoted by (ϕ, pred), where $\phi \neq \phi^*$. On receiving (ϕ, pred), the challenger runs CDecap on input (ϕ, pred) and passes the resulting decapsulation to $\mathcal{A}$. At the end of this phase, $\mathcal{A}$ outputs its guess $\beta' \in \{0,1\}$.
We define the attacker $\mathcal{A}$'s success probability by

$$\mathbf{Adv}^{\mathrm{CCCA}}_{\mathcal{A},\mathsf{KEM}}(\lambda) = \Big| \Pr[\beta' = \beta] - \frac{1}{2} \Big|.$$

Notice that this notion becomes CCA (for KEM) when "pred" always outputs 1 for any input but is equivalent to the confidentiality against passive attack when "pred" always outputs 0 for any input. Hence, in general, CCCA is weaker than CCA. Indeed, admissible predicates must satisfy that the amount of uncertainty the attacker has about the key is negligible. Formally, let $Q_{\mathcal{A}}$ denote the number of decapsulation queries made by the attacker $\mathcal{A}$. Then for attacker $\mathcal{A}$ in the above experiment we define plaintext uncertainty $\mathrm{uncert}_{\mathcal{A}}(\lambda)$ as follows:

$$\mathrm{uncert}_A(\lambda) = \frac{1}{Q_{\mathcal{A}}} \sum_{1 \le i \le Q_{\mathcal{A}}} \Pr[\mathrm{pred}_i(K) = 1 \mid K \xleftarrow{\$} S_K]$$

where $\mathrm{pred}_i : S_K \to \{0,1\}$ is the predicate $\mathcal{A}$ submits with the i-th decapsulation query. An attacker $\mathcal{A}$ is said to be "valid" if $\mathrm{uncert}_{\mathcal{A}}(\lambda)$ is negligible in λ. We say

that KEM is CCCA-secure if $\mathbf{Adv}^{\mathrm{CCCA}}_{\mathsf{KEM}}(\lambda) = \max_{\mathcal{A}} \left\{\mathbf{Adv}^{\mathrm{CCCA}}_{\mathcal{A},\mathsf{KEM}}(\lambda)\right\}$ is negligible for any admissible attacker $\mathcal{A}$.

We remark that before the CCCA notion was defined, Abe et al. [2] had defined a very similar notion called "LCCA". In LCCA, an attacker also has access to the "restricted decapsulation oracle", which outputs a key (decapsulation) of a queried ciphertext only when a certain predicate taking the key as input returns 1. They showed that the KEM part of Kurosawa and Desmedt's hybrid encryption scheme is actually LCCA-secure. We note that our KEM construction based on the CCCA-secure KEM, which will be presented in the next section, is still secure even if the underlying KEM is assumed to be LCCA-secure. However, due to its rigor (e.g. precise definition of uncertainty), we use the CCCA notion.

2.2 Message Authentication Code and Key Derivation Function

Message Authentication Code. Let $\mathsf{MAC} = (\mathsf{MAC.Sign}, \mathsf{MAC.Ver})$ be a MAC (Message Authentication Code) scheme whose key space S_κ is defined by the security parameter λ. On input (κ, m), where $\kappa(\in S_\kappa)$ and m denote a MAC key and a message resp., $\mathsf{MAC.Sign}$ produces a MAC (tag) σ. On input (κ, σ, m), $\mathsf{MAC.Ver}$ outputs 1 if (σ, m) is valid with respect to κ, or outputs 0, otherwise.

In the literature, there exist several unforgeability notions for MAC. The one we need in this paper is "strong unforgeability under chosen message attack (SUF-CMA)" [5], which can be defined as follows. Given access to the oracle $\mathsf{MAC.Sign}(\kappa, \cdot)$ where κ is chosen uniformly at random from S_κ, an attacker $\mathcal{A}$ outputs (m, σ) such that $\mathsf{MAC.Ver}(\kappa, \sigma, m) = 1$ and σ was never returned by $\mathsf{MAC.Sign}(\kappa, \cdot)$. We define the attacker $\mathcal{A}$'s success probability by $\mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(\lambda)$. As usual, we say that MAC is SUF-CMA secure if $\mathbf{Adv}^{\text{SUF-CMA}}_{\mathsf{MAC}}(\lambda) = \max_{\mathcal{A}} \left\{\mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{A},\mathsf{MAC}}(\lambda)\right\}$ is negligible for any admissible attacker $\mathcal{A}$.

Though this notion seems stronger than usual UF-CMA notion for MAC, as pointed out in [5], any pseudorandom function (PRF) is a SUF-CMA secure MAC, and many practical MAC schemes, e.g., HMAC and CBC-MAC are actually SUF-CMA secure.

Key Derivation Function. Also, we will need the key derivation function [9,13], denoted KDF, for our construction of the KEM scheme. KDF satisfies the following security requirement, called "real or random (ROR)". Assume that KDF takes an element a chosen uniformly at random from the appropriate domain S_a. Let l be the length of the output of KDF, which depends on the security parameter λ. We define the security of KDF in the ROR sense, with respect to an attacker $\mathcal{A}$, as follows.

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{A},\mathsf{KDF}}(\lambda) = |\,&\Pr[a \xleftarrow{\$} S_a : 1 \leftarrow \mathcal{A}(1^\lambda, \mathsf{KDF}(a))] \\ &- \Pr[a \xleftarrow{\$} S_a; \mu \xleftarrow{\mathrm{R}} \{0,1\}^l : 1 \leftarrow \mathcal{A}(1^\lambda, \mu)]|.\end{aligned}$$

We say that KDF is ROR-secure if $\mathbf{Adv}^{\mathrm{ROR}}_{\mathsf{KDF}}(\lambda) = \max_{\mathcal{A}} \left\{\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{A},\mathsf{KDF}}(\lambda)\right\}$ is negligible for any admissible attacker $\mathcal{A}$.

3 Our Construction of CCA-Secure KEM from CCCA-Secure KEM

Description. Let $\mathsf{KEM}' = (\mathsf{KEM}'.\mathsf{Gen}, \mathsf{KEM}'.\mathsf{Encap}, \mathsf{KEM}'.\mathsf{Decap})$ be a KEM scheme. Let $\mathsf{MAC} = (\mathsf{MAC.Sign}, \mathsf{MAC.Ver})$ be a MAC scheme; and let $\mathsf{KDF} : S_{\tilde{K}} \rightarrow S_K \times S_\kappa$ be a key derivation function, where $S_{\tilde{K}}$ denotes the key space of KEM'; S_K denotes the key space of the resulting KEM scheme KEM (i.e., the key space of a given DEM scheme); and S_κ denotes the key space of the MAC scheme MAC. Based on these primitives, we construct a new KEM scheme $\mathsf{KEM} = (\mathsf{KEM.Gen}, \mathsf{KEM.Encap}, \mathsf{KEM.Decap})$ as described in Figure 1.

$\mathsf{KEM.Gen}(1^\lambda)$	$\mathsf{KEM.Encap}(pk)$	$\mathsf{KEM.Decap}(sk, \phi)$
$(pk', sk') \leftarrow \mathsf{KEM}'.\mathsf{Gen}(1^\lambda)$	$pk \xrightarrow{\text{par}} (pk', \mathsf{KDF}, \mathsf{MAC})$	$sk \xrightarrow{\text{par}} sk'$
Select KDF and MAC	$(\theta, \tilde{K}) \leftarrow \mathsf{KEM}'.\mathsf{Encap}(pk')$	$\phi \xrightarrow{\text{par}} (\theta, \sigma)$
$pk \leftarrow (pk', \mathsf{KDF}, \mathsf{MAC})$	$(K, \kappa) \leftarrow \mathsf{KDF}(\tilde{K})$	$\tilde{K} \leftarrow \mathsf{KEM}'.\mathsf{Decap}(sk', \theta)$
$sk \leftarrow sk'$	$\sigma \leftarrow \mathsf{MAC.Sign}(\kappa, \theta)$	$(K, \kappa) \leftarrow \mathsf{KDF}(\tilde{K})$
Return (pk, sk)	$\phi \leftarrow (\theta, \sigma)$	If $\mathsf{MAC.Ver}(\kappa, \sigma, \theta) = 1$
	Return (ϕ, K)	then return K
		Else return $\perp$

Fig. 1. Our Generic Construction of CCA-Secure KEM from CCCA-Secure KEM

We note that the above technique of applying a MAC to a ciphertext has widely been used in the literature. For example, Boneh and Katz [7] used a similar technique to convert an identity-based encryption scheme to a CCA-secure (normal) public key encryption scheme. Another example is "Encrypt-then-MAC" construction of authenticated encryption given in [5]. Our construction shows that this powerful technique also can yield a conversion from CCCA-secure KEM to CCA-secure KEM.

Security Analysis. We show that the generic construction of KEM presented in Figure 1 is CCA-secure assuming that the underlying KEM', MAC and KDF schemes are secure in the sense defined in the previous section. More precisely, we prove the following theorem.

Theorem 1. *If* KEM' *is CCCA-secure;* MAC *is SUF-CMA secure; and* KDF *is ROR-secure, the scheme* KEM *described in Figure 1 is CCA-secure. That is, we obtain the following bounds:*

$$\mathbf{Adv}^{CCA}_{\mathsf{KEM}}(\lambda) \leq 4\mathbf{Adv}^{CCCA}_{\mathsf{KEM}'}(\lambda) + 5\mathbf{Adv}^{ROR}_{\mathsf{KDF}}(\lambda) + q_D\mathbf{Adv}^{SUF\text{-}CMA}_{\mathsf{MAC}}(\lambda),$$

where λ *denotes the security parameter and* q_D *denotes the number of decapsulation queries.*

The proof is given in Appendix A.

4 Applications of Our KEM Construction

4.1 New CCA-Secure KEMs from Well-Known CCCA-Secure KEMs

Using our method that transforms CCCA-secure KEM to CCA-secure KEM, one obtains new CCA-secure KEM schemes from the well-known CCCA-secure KEM schemes in the literature, the Kurosawa-Desmedt KEM [17] and the Hofheinz-Kiltz KEM [15] schemes.

First we describe the new KEM scheme KDKEM, which is constructed using the Kurosawa-Desmedt KEM scheme. Let $\mathbb{G}$ be a finite group, generated by g_1 and g_2. Assume that the order of this group is p, a prime, and that DDH problem is hard in this group. (The formal definition of the DDH problem can be found in Appendix B.) Also, assume that the key derivation function KDF and the MAC scheme MAC satisfy the security requirements described in Section 2.2. Let TCR denote a target collision resistant hash function as defined in [17]. In Figure 2, we describe each sub-algorithm of KDKEM.

$\mathsf{KDKEM.Gen}(1^\lambda)$	$\mathsf{KDKEM.Encap}(pk)$	$\mathsf{KDKEM.Decap}(sk, \phi)$
Select KDF, MAC, TCR	$r \xleftarrow{\$} \mathbb{Z}_p^*;\ u_1 \leftarrow g_1^r;\ u_2 \leftarrow g_2^r$	$\phi \xrightarrow{\text{par}} (u_1, u_2, \sigma)$
$x_1, x_2, y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p^*$	$\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$	$\alpha \leftarrow \mathsf{TCR}(u_1, u_2)$
$c \leftarrow g_1^{x_1} g_2^{x_2};\ d \leftarrow g_1^{y_1} g_2^{y_2}$	$\theta \leftarrow (u_1, u_2) \in \mathbb{G}^2$	$\tilde{K} \leftarrow u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha}$
$pk' \leftarrow (\lambda, p, g_1, g_2, c, d)$	$\tilde{K} \leftarrow c^r d^{r\alpha} \in \mathbb{G}$	$(K, \kappa) \leftarrow \mathsf{KDF}(\tilde{K})$
$sk' \leftarrow (x_1, x_2, y_1, y_2)$	$(K, \kappa) \leftarrow \mathsf{KDF}(\tilde{K})$	If $\mathsf{MAC.Ver}(\kappa, \sigma, (u_1, u_2)) = 1$
$pk \leftarrow (pk', \mathsf{KDF}, \mathsf{MAC})$	$\sigma \leftarrow \mathsf{MAC.Sign}(\kappa, \theta)$	return K
$sk \leftarrow sk'$	$\phi \leftarrow (\theta, \sigma)$	Else return $\perp$
Return (sk, pk)	Return (ϕ, K)	

Fig. 2. CCA-Secure KEM from Kurosawa-Desmedt CCCA-Secure KEM [17]

Recall that according to Theorem 1, if the Kurosawa-Desmedt KEM scheme is CCCA-secure, the above KDKEM scheme is CCA-secure. In [15], the Kurosawa-Desmedt KEM scheme is shown to be CCCA-secure assuming that the DDH problem is hard. Thus the KDKEM scheme is CCA-secure.

Another KEM scheme HKKME which is constructed using the Hofheinz-Kiltz KEM scheme [15] can be described as follows. Let $\mathbb{G}$ be a finite group as defined previously. Let g be a generator of $\mathbb{G}$. Assume that KDF and MAC are as defined in Section 2.2. Figure 3 describes each sub-algorithm of the scheme HKKEM.

Since the the Hofheinz-Kiltz KEM scheme is shown to be CCCA-secure assuming that the DDH problem is hard, the HKKEM scheme presented in Table 3 is CCA-secure.

We remark that the random oracle model is not required to analyze the two KEM schemes presented above.

$\mathsf{HKKEM.Gen}(1^\lambda)$	$\mathsf{HKKEM.Encap}(pk)$	$\mathsf{HKKEM.Decap}(sk, \phi)$
Select KDF, TCR, MAC	$r \xleftarrow{\$} \mathbb{Z}_p^*$; $c \xleftarrow{\$} g^r$	$\phi \xrightarrow{\text{par}} (c, \pi, \sigma)$
$x_1, y_1, y_2, w \xleftarrow{\$} \mathbb{Z}_p^*$	$t \leftarrow \mathsf{TCR}(c)$; $\pi \leftarrow (u^t v)^r$	If $c \notin \mathbb{G}$ or $\pi \notin \mathbb{G}$ return $\perp$
$h \leftarrow g^w$; $u \leftarrow g^{-x_1/y_2}$	$\theta \leftarrow (c, \pi) \in \mathbb{G}^2$	$t \leftarrow \mathsf{TCR}(c)$
$v \leftarrow g^{-y_1/y_2} h^{1/y_2}$	$\tilde{K} \leftarrow h^r \in \mathbb{G}$	If $c^{xt+y} \neq \pi$ return $\perp$
$pk' \leftarrow (u, v) \in \mathbb{G}^2$	$(K, \kappa) \leftarrow \mathsf{KDF}(\tilde{K})$	$\tilde{K} \leftarrow c^{x_1 t + y_1} \pi^{y_2}$
$sk' \leftarrow (x_1, y_1, y_2) \in \mathbb{Z}_p^3$	$\sigma \leftarrow \mathsf{MAC.Sign}(\kappa, \theta)$	$(K, \kappa) \leftarrow \mathsf{KDF}(\tilde{K})$
$pk \leftarrow (pk', \mathsf{KDF}, \mathsf{MAC})$	$\phi \leftarrow (\theta, \sigma)$	If $\mathsf{MAC.Ver}(\kappa, \sigma, (c, \pi)) = 1$
$sk \leftarrow sk'$	Return (ϕ, K)	return K
Return (sk, pk)		Else return $\perp$

Fig. 3. CCA-Secure KEM Constructed from Hofheinz-Kiltz CCCA-Secure KEM [15]

4.2 Efficiency Comparisons with Other Schemes

We now compare the KDKEM and HKKEM schemes with other well-known KEM schemes.

First, we compare the length of ciphertext, the cost for encapsulation and decapsulation in KDKEM and HKKEM with those of the KEM schemes constructed by choosing a key K uniformly at random (from an appropriate DEM-key space) and encrypting it with CCA-secure hybrid encryption schemes by Kurosawa and Desmedt [13] and Hofheinz and Kiltz [15]. We denote the two KEM schemes constructed in this way by $\mathsf{KDHE2KEM}$ and $\mathsf{HKHE2KEM}$ respectively. (In [2], it is shown that these KEM schemes are CCA-secure.) We also compare the length of ciphertext and the cost for encapsulation/decapsulation of our schemes with those of Cramer and Shoup's [9] KEM scheme, named "$\mathsf{CS3}$". Note that the security of all the schemes is based on the DDH problem and does not depend on random oracles. In Table, 1, we summarize the comparisons.

Next, we compare the length of ciphertext and the cost for encryption and decryption of the hybrid encryption schemes constructed by combining our two CCA-secure KEM schemes with *length-preserving* one-time CCA-secure DEMs [21] (following the KEM/DEM framework) with those of Kurosawa and Desmedt's [13] CCA-secure hybrid encryption scheme; Hofheinz and Kiltz's [15] CCA-secure hybrid encryption scheme; and the hybrid encryption schemes constructed by combining length-preserving CCA-secure DEMs with Cramer and Shoup's [9] KEM scheme. We assume that the length of the plaintext to be encrypted is the same for all the schemes. We assume that the same MAC algorithm is used. In Table 2, we summarize the comparisons.

We observe that compared to the previous CCA-secure KEM schemes based on the DDH problem, the KEM schemes obtained by our proposed transformation (Figures 2 and 3) satisfy that either:

- are more computationally efficient (saving one exponentiation in encapsulation/decapsulation) or
- have shorter ciphertext (about 80 bits).

Table 1. Efficiency comparison for DDH-based CCA-secure KEMs in the standard model. KDHE2KEM and HKHE2KEM are the KEM schemes constructed by choosing a key K uniformly at random and encrypting it with CCA-secure hybrid encryption schemes by Kurosawa and Desmedt [13], and Hofheinz and Kiltz [15] resp. CS3 is the CCA-secure KEM proposed by Cramer& Shoup in [9]. $|p|$ is the bit-length of the representation of an element in $\mathbb{G}$. $|\mathrm{enc}|$ denotes the output bit-length of a block cipher used to encrypt a random key K, and $|\sigma|$ denotes the length of the output of a MAC. macG denotes the MAC generation operation, and macV denotes the MAC verification operation. In practice, $|\mathrm{enc}| = 128$ (cf. [10]), and for the current security level (i.e. $\lambda = 2^{80}$), $|\sigma| = 80$ and $|p| = 160$, and the relative timing for the operations are multi-exponentiation ≈ 1.5, regular exponentiation $= 1$.

Scheme	Ciphertext Size	Encapsulation	Decapsulation
		#[multi,regular]-exp + extra operation	
KDKEM (Figure 2)	$2\|p\| + \|\sigma\|$	$[1, 2] + \mathrm{macG}$	$[1, 0] + \mathrm{macV}$
HKKEM (Figure 3)	$2\|p\| + \|\sigma\|$	$[1, 2] + \mathrm{macG}$	$[1, 0] + \mathrm{macV}$
KDHE2KEM	$2\|p\| + \|\mathrm{enc}\| + \|\sigma\|$	$[1, 2] + \mathrm{macG}$	$[1, 0] + \mathrm{macV}$
HKHE2KEM	$2\|p\| + \|\mathrm{enc}\| + \|\sigma\|$	$[1, 2] + \mathrm{macG}$	$[1, 0] + \mathrm{macV}$
CS3	$3\|p\|$	$[1, 3]$	$[1, 1]$

Table 2. Efficiency comparison for DDH-based CCA-secure hybrid encryption schemes in the standard model. lpDEM denotes a length-preserving CCA-secure DEM [21]. KDHE and HKHE denote the hybrid encryption schemes proposed by Kurosawa & Desmedt [13] and Hofheinz & Kiltz [15] resp. $|p|$ is the bit-length of the representation of an element in $\mathbb{G}$. $|\sigma|$ is the bit-length of the output of a MAC. $|\sigma|$ is the bit-length of the output of a MAC. $|m|$ is the bit-length of the plaintext message m. macG denotes the MAC generation operation, and macV denotes the MAC verification operation. enc and dec denote the block cipher encryption and decryption operations resp. For comparison and for the current security level (i.e. $\lambda = 2^{80}$), tag $= 80$ and $|p| = 160$, and the relative timing for the operations are multi-exponentiation ≈ 1.5, regular exponentiation $= 1$.

Scheme	Ciphertext Size	Encryption	Decryption
		#[multi,regular]-exp + extra operations	
KDKEM + lpDEM	$2\|p\| + \|\sigma\| + \|m\|$	$[1, 2] + \mathrm{macG} + \mathrm{enc}$	$[1, 0] + \mathrm{macV} + \mathrm{dec}$
HKKEM + lpDEM	$2\|p\| + \|\sigma\| + \|m\|$	$[1, 2] + \mathrm{macG} + \mathrm{enc}$	$[1, 0] + \mathrm{macV} + \mathrm{dec}$
KDHE [13]	$2\|p\| + \|\sigma\| + \|m\|$	$[1, 2] + \mathrm{macG} + \mathrm{enc}$	$[1, 0] + \mathrm{macV} + \mathrm{dec}$
HKHE [15]	$2\|p\| + \|\sigma\| + \|m\|$	$[1, 2] + \mathrm{macG} + \mathrm{enc}$	$[1, 0] + \mathrm{macV} + \mathrm{dec}$
CS3 + lpDEM	$3\|p\| + \|m\|$	$[1, 3] + \mathrm{enc}$	$[1, 1] + \mathrm{dec}$

Interestingly, by plugging the KEMs KDKEM and HKKEM using into the KEM/DEM framework, one obtains hybrid encryption schemes which are as efficient as the Kurosawa&Desmedt and Hofheinz&Kiltz encryption schemes. This shows it is possible to transfer the efficiency of the latter schemes to the original KEM/DEM framework.

5 Concluding Remarks

In this paper, we presented a construction of CCA-secure (strongly secure) KEM from CCCA-secure (weakly secure) KEM. Using our construction, we built concrete CCA-secure KEM schemes from Kurosawa and Desmedt's [17], and Hofheinz and Kilt's [15] CCCA-secure KEM schemes. We showed that when our CCA-secure KEM schemes are combined with length-preserving one-time CCA-secure DEMs, one can obtain hybrid encryption schemes *as efficient as* the hybrid encryption schemes proposed in [13,15]. Compared with the schemes in [13,15], a very interesting feature of our schemes is that their CCA-security can be explained using the original KEM/DEM framework (while the security of the aforementioned schemes falls outside the KEM/DEM framework.)

Acknowledgements

The authors are grateful to the anonymous referees of SCN '08 for their helpful comments. The first and fourth authors are partially supported by the European Union project SMEPP-033563. The second author acknowledges the support of the Spanish *Ministerio de Educación y Ciencia* under the project ARES (Consolider Ingenio 2010 CSD2007-00004). The third author is partially supported by ARC Discovery Grant DP0877123.

References

1. Abe, M., Genaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
2. Abe, M., Genaro, R., Kurosawa, K.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM, Cryptology ePrint Archive, Report 2005/027 (2005) (Last update: 11 October 2006)
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
5. Bellare, M., Namprepre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM-CCS 1993, pp. 62–73. ACM, New York (1993)
7. Boneh, D., Katz, J.: Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
8. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)

9. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal of Computing 33, 167–226 (2003)
10. Cramer, R., Shoup, V.: Signature Schemes Based on the Strong RSA Assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)
11. Dent, A.: Hybrid Cryptography, Cryptology ePrint Archive, Report 2004/210 (2004)
12. Dolev, D., Dwork, C., Naor, M.: Non-malleable Cryptography. In: STOC 1991, pp. 542–552. ACM, New York (1991)
13. Gennaro, R., Shoup, V.: A Note on An Encryption Scheme of Kurosawa and Desmedt, Cryptology ePrint Archive, Report 2004/294 (2004)
14. Herranz, J., Hofheinz, D., Kiltz, E.: The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure,Cryptology ePrint Archive, Report 2006/207 (2006)
15. Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
16. ISO 18033-2, An Emerging Standard for Public-Key Encryption (2004)
17. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
18. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17(2), 373–386 (1988)
19. Naor, M., Yung, M.: Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In: STOC 1990, pp. 427–437. ACM, New York (1990)
20. Okamoto, T.: Authenticated Key Exchange and Key Encapsulation in the Standard Model. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 474–484. Springer, Heidelberg (2007)
21. Phan, D., Pointcheval, D.: About the security of ciphers (semantic security and pseudo-random permutations). In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 182–197. Springer, Heidelberg (2004)
22. V. Shoup, A Proposal for an ISO Standard for Public Key Encryption (version 2.1), ISO/IEC JTC 1/SC 27 (2001)

A Proof of Theorem 1

Proof. The proof essentially follows the logic of the proof for the Tag-KEM construction given in Section 4.3 of [2]. However, a sizable difference is that 1) since the "Tag" τ in the Tag-KEM construction of [2] is not used any more, the MAC scheme in our KEM construction should be treated differently (Indeed, our construction requires the MAC to be SUF-CMA secure, which is different from the security requirement of the MAC in [2].); and 2) we need to handle CCCA-security of KEM as the underlying primitive.

– Game $\mathsf{G_0}$: This game is identical to the IND-CCA game described in Section 2. We repeat this game to clear up the notations. Let $\mathcal{A}$ be an IND-CCA attacker for the scheme KEM. Suppose that pk and sk be the public and private keys resp., to which $\mathcal{A}$ has access. We denote the ith ciphertext under consideration by $\phi_i = (\theta_i, \sigma_i)$, where $i \in \{1, \ldots, q_D\}$. We denote a challenge ciphertext by $\phi^* = (\theta^*, \sigma^*)$, which satisfies the following implicit relations:

$$(K_1^*, \kappa_1^*) = \mathsf{KDF}(\tilde{K}_1^*);\ \sigma^* = \mathsf{MAC}(\kappa_1^*, \theta^*) \text{ where } \tilde{K}_1^* = \mathsf{KEM}'.\mathsf{Decap}(sk, \theta^*).$$

We denote a key chosen randomly from S_K by K_0^*. Note here that the challenger picks $\beta \in \{0,1\}$ at random and returns (ϕ^*, K_β^*) to $\mathcal{A}$.
We denote by S_0 the event $\beta' = \beta$, where β' is a bit output by A at the end of the game. (We use a similar notation $S_1, S_2, \ldots$ for all modified games $\mathsf{G}_1, \mathsf{G}_2, \ldots$ respectively).

– Game G_1: In this game, we modify the generation of the challenge ciphertext in such a way that when $\beta = 0$, we choose $\tilde{K}_0^*$ at random from $S_{\tilde{K}}$ and compute $(K_0^*, \mu) \leftarrow \mathsf{KDF}(\tilde{K}_0^*)$, where $\mu \in S_\kappa$ is an arbitrary string.
Hence we get

$$|\Pr[S_0] - \Pr[S_1]| = \frac{1}{2}|\Pr[S_0|\beta = 0] - \Pr[S_1|\beta = 0]| \leq \mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda),$$

for some $\mathcal{B}$ that breaks the ROR-security of KDF.

– Game G_2: In this game, we add another special rejection rule called SR2 to the decapsulation algorithm of the previous game.

SR2 If a ciphertext $\phi_i = (\theta_i, \sigma_i)$ such that $\theta_i = \theta^*$ is submitted in Phase 2 of the IND-CCA game then immediately output $\perp$.

Let F_2 be an event that a ciphertext (submitted to the decapsulation oracle) is rejected under the rule of SR2 but would have been accepted under the rule of Game G_2. Since Game G_1 and Game G_2 proceed identically until this event happens and in particular, $S_1 \wedge \neg F_2$ and $S_2 \wedge \neg F_2$ are identical. Thus we have

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[F_2].$$

Now, we construct an attacker $\mathcal{C}$ that breaks IND-CCCA of the scheme KEM' using $\mathcal{A}$ as subroutine as follows. Assume that $\mathcal{C}$ is provided with $(pk', \theta^*, \tilde{K}_b^*)$ where b is a random bit and that $\mathcal{C}$ defines the function predicate pred as

$$\mathrm{pred}(\tilde{K}) = \begin{cases} 1 & : \text{ if } \mathsf{MAC.Ver}(\kappa, \sigma, \theta) = 1 \\ 0 & : \text{ otherwise} \end{cases}$$

where $\tilde{K} = \mathsf{KEM}'.\mathsf{Decap}(sk, \theta)$ and $(K, \kappa) = \mathsf{KDF}(\tilde{K})$. Note here that $\sigma \in R_{\mathsf{MAC}}$, where R_{MAC} denotes the range of the MACs, is hard-coded into $\mathrm{pred}(\cdot)$. Below, we describe a complete specification of $\mathcal{C}$.

Algorithm $\mathcal{C}(pk', \theta^*, \tilde{K}_b^*)$
  $pk \leftarrow (pk', \mathsf{KDF}, \mathsf{MAC})$; Send pk to $\mathcal{A}$
  $(K^*, \kappa^*) \leftarrow \mathsf{KDF}(\tilde{K}_b^*)$; $\sigma^* \leftarrow \mathsf{MAC}(\kappa^*, \theta^*)$
  $\phi^* \leftarrow (\theta^*, \sigma^*)$; Send (ϕ^*, K^*) to $\mathcal{A}$
  If $\mathcal{A}$ submits $\phi_i = (\theta_i, \sigma_i)$ to the decapsulation oracle
    Submit $(\theta_i, \mathrm{pred}_i)$ to its CDecap oracle
    If $\theta_i = \theta^*$ then
      If CDecap does not reject[5] then return $b' = 1$ and halt

[5] Note that σ_i must be different from σ^* because by definition $\mathcal{A}$ is not allowed to issue the challenge ciphertext (θ^*, σ^*) as a decapsulation query.

Else send $\perp$ to $\mathcal{A}$ and continue
Else
Get $\tilde{K}_i$ or $\perp$ from CDecap and simulate the decapsulation oracle of $\mathcal{A}$ if $\tilde{K}_i$ is returned or send $\perp$ to $\mathcal{A}$
If $\mathcal{A}$ stops, output $b' = 0$

First, assume that $b = 1$ in the above construction. In this case, K^* and κ^* are generated from $\tilde{K}_1^*$, i.e., $K^* = K_\beta^*$ and $\kappa^* = \kappa_\beta^*$ when $\beta = 1$. Now consider the event $b' = 1$. By the construction of $\mathcal{C}$ given above, $b' = 1$ happens when the decapsulation query (θ^*, σ_i) (such that $\sigma_i \neq \sigma^*$) is not rejected. But according to SR2, this query should be rejected straight away. That is, the event $b' = 1$ when $b = 1$ is equivalent to the event F_2 when $\beta = 1$. Hence, we have $\Pr[F_2|\beta = 1] = \Pr[b' = 1|b = 1]$.

Now, assume that $b = 0$ in the above construction. Let F_2' represent the event that $b' = 1$ when $b = 0$. We then modify the above construction of $\mathcal{C}$ so that (K^*, κ^*) is selected independently and uniformly at random from $S_K \times S_\kappa$. Let F_2'' be an event that $b' = 1$ under this modification. It is easy to see that $|\Pr[F_2'] - \Pr[F_2'']| \leq 2\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda)$ for some attacker $\mathcal{B}$ that breaks the ROR-security of KDF. Now, pick $j \in \{1, \ldots, q_D\}$ at random and define F_2''' be an event that for the jth ciphertext ψ_j (submitted to the decapsulation oracle), $b' = 1$. Note that we have $\Pr[F_2''] \leq q_D \Pr[F_2''']$. We now prove the following claim.

Claim. $\Pr[F_2'''] \leq \mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda)$.

Proof. Let $\mathcal{D}$ be a SUF-CMA attacker for MAC. Assume that $\mathcal{D}$ is given access to the MAC-generation oracle $\mathsf{MAC.Sign}(\kappa^*, \cdot)$, where κ^* is chosen uniformly at random from S_κ. First, $\mathcal{D}$ generates (sk, pk) and gives pk to $\mathcal{A}$. $\mathcal{D}$ then parses pk as $(pk', \mathsf{KDF}, \mathsf{MAC})$, computes $(\theta^*, \tilde{K}) \leftarrow \mathsf{KEM'.Encap}(pk')$. $\mathcal{D}$ queries θ^* to its MAC-generation oracle to get $\sigma^* = \mathsf{MAC.Sign}(\kappa^*, \theta^*)$. $\mathcal{D}$ gives (θ^*, σ^*) as a challenge ciphertext to $\mathcal{A}$. (Note that (θ^*, σ^*) is identically distributed as the challenge ciphertext that $\mathcal{A}$ is given from the above *modified* $\mathcal{C}$ where κ^* is chosen independently and uniformly at random.) $\mathcal{D}$ responds to A's decapsulation queries using sk. When $\mathcal{A}$ submits (θ^*, σ_i) such that $\mathsf{MAC.Ver}(\kappa^*, \sigma_i, \theta^*) = 1$, $\mathcal{D}$ outputs σ_i as a forgery. (Note that σ_i was never returned by $\mathcal{D}$'s MAC-generation oracle.) Since the event $b' = 1$ occurs when (θ^*, σ_i) is not rejected, i.e. $\mathsf{MAC.Ver}(\kappa^*, \sigma_i, \theta^*) = 1$, we have $\Pr[F_2'''] \leq \mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda)$.

Consequently, we obtain

$$\Pr[F_2'] = \Pr[b' = 1|b = 0] \leq 2\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda) + q_D\mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda).$$

Note that by definition $\frac{1}{2}|\Pr[b'=1|b=1] - \Pr[b'=1|b=0]| = \mathbf{Adv}^{\text{IND-CCCA}}_{\mathcal{C},\mathsf{KEM'}}(\lambda)$. Thus, we get

$$\Pr[F_2|\beta = 1] \leq 2\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda) + 2\mathbf{Adv}^{\text{IND-CCCA}}_{\mathcal{C},\mathsf{KEM'}}(\lambda) + q_D\mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda).$$

Next, consider the case $\beta = 0$. Recall that by the rule set in Game G_1, when $\beta = 0$, $\tilde{K}_0^*$ is chosen at random from $S_{\tilde{K}}$ and K_0^* is obtained by computing $(K_0^*, \mu) \leftarrow \mathsf{KDF}(\tilde{K}_0^*)$ for some arbitrary string $\mu \in S_\kappa$. We now slightly modify the above algorithm $\mathcal{C}$ in such a way that K^* is (always) obtained by computing $(K^*, \mu) \leftarrow \mathsf{KDF}(\tilde{K}_0^*)$, where $\tilde{K}_0^*$ is chosen uniformly at random from $S_{\tilde{K}}$. That is, K^* is distributed independently from other variables. Hence, we have $\Pr[F_2|\beta = 0] = \Pr[b' = 1|b = 1]$.

Also, we can show in exactly the same way as done before that

$$\Pr[b' = 1|b = 0] \leq 2\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda) + q_D\mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda)$$

Consequently, we have

$$\Pr[F_2] \leq 2\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda) + 2\mathbf{Adv}^{\text{IND-CCCA}}_{\mathcal{C},\mathsf{KEM}'}(\lambda) + q_D\mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda).$$

– Game G_3: In this game, we modify the generation of the challenge ciphertext in such a way that κ_1^* is obtained by computing $(\nu, \kappa_1^*) \leftarrow \mathsf{KDF}(\tilde{K}_\beta^*)$ for arbitrary $\nu \in S_K$. Hence, when $\beta = 1$, the views of $\mathcal{A}$ in Game G_3 and Game G_2 are identically distributed. Thus we get

$$\begin{aligned}|\Pr[S_2]\text{–}\Pr[S_3]| &= \frac{1}{2}|\Pr[S_2|\beta=1]+\Pr[S_2|\beta=0]-\Pr[S_3|\beta=1]-\Pr[S_3|\beta=0]| \\ &= \frac{1}{2}|\Pr[S_2|\beta=0]-\Pr[S_3|\beta=0]|.\end{aligned}$$

The above equality implies that we only need to consider the case $\beta = 0$.

Now we construct again attacker $\mathcal{C}$ that breaks IND-CCCA of KEM' using $\mathcal{A}$ as subroutine: As with the case of $\mathcal{C}$, we assume that $\mathcal{C}$ is provided with $(pk', \theta^*, \tilde{K}_b^*)$ where b is a random bit and that $\mathcal{C}$ defines the function predicate pred as

$$\mathrm{pred}(\tilde{K}) = \begin{cases} 1 & : \text{ if } \mathsf{MAC.Ver}(\kappa, \sigma, \theta) = 1 \\ 0 & : \text{ otherwise} \end{cases}$$

where $\tilde{K} = \mathsf{KEM}'.\mathsf{Decap}(sk, \theta)$ and $(K, \kappa) = \mathsf{KDF}(\tilde{K})$. Note here that $\sigma \in R_{\mathsf{MAC}}$, where R_{MAC} denotes the range of the MACs, is hard-coded into $\mathrm{pred}(\cdot)$.

Algorithm $\mathcal{C}(pk', \theta^*, \tilde{K}_b^*)$

$pk \leftarrow (pk', \mathsf{KDF}, \mathsf{MAC})$; Send pk to $\mathcal{A}$; $\tilde{K'}_0^* \xleftarrow{\$} S_{\tilde{K}}$
$(K^*, \mu) \leftarrow \mathsf{KDF}(\tilde{K'}_0^*)$; $(\nu, \kappa^*) \leftarrow \mathsf{KDF}(\tilde{K}_b^*)$; $\sigma^* \leftarrow \mathsf{MAC}(\kappa^*, \theta^*)$
$\phi^* \leftarrow (\theta^*, \sigma^*)$; Send (ϕ^*, K^*) to $\mathcal{A}$
If $\mathcal{A}$ submits $\phi_i = (\theta_i, \sigma_i)$ to the decapsulation oracle
  Submit $(\theta_i, \mathrm{pred}_i)$ to its CDecap oracle
  If $\theta_i = \theta^*$ then return $\perp$
  Else
    Get $\tilde{K}_i$ or $\perp$ from CDecap and simulate the decapsulation oracle of $\mathcal{A}$ if $\tilde{K}_i$ is returned or send $\perp$ to $\mathcal{A}$
If $\mathcal{A}$ outputs β', output it as b'

First, assume that $b = 1$ in the above construction. In this case, σ^* is generated from the correct key $\tilde{K}_1^*$ while K^* is computed from the key $\tilde{K'}_0^*$ chosen randomly from $S_{\tilde{K}}$. Hence the view of $\mathcal{A}$ simulated by the algorithm $\mathcal{B}_4$ is the same as that of $\mathcal{A}$ in Game $\mathsf{G_2}$ at $\beta = 0$. Thus, we have $\Pr[S_2|\beta = 0] = \Pr[b' = 0|b = 1]$.

Now, assume that $b = 0$ in the above construction. In this case, σ^* is generated using $\tilde{K}_0^*$ chosen randomly from $S_{\tilde{K}}$. Notice that the view of A in the above simulation is almost the same as that in this game (Game $\mathsf{G_3}$) when $\beta = 0$ except that in this game, σ^* and K^* are generated from the single input to KDF while they are independently generated in the above simulation. However, by the result of [2], we get $|\Pr[S_3|\beta = 0] - \Pr[b' = 0|b = 0]| \leq 4\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda)$.

Putting all the bounds together, we get

$$\begin{aligned}|\Pr[S_2] - \Pr[S_3]| &= \frac{1}{2}|\Pr[S_2|\beta = 0] - \Pr[S_3|\beta = 0]| \\ &\leq \frac{1}{2}|\Pr[b' = 0|b = 1] - \Pr[b' = 0|b = 0] - 4\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda)| \\ &\leq \mathbf{Adv}^{\text{IND-CCCA}}_{\mathcal{C},\mathsf{KEM'}}(\lambda) + 2\mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda).\end{aligned}$$

Observe that we can build up $\mathcal{C}$ again using $\mathcal{A}$ in this game (Game $\mathsf{G_3}$).

Algorithm $\mathcal{C}(pk', \theta^*, \tilde{K}_b^*)$
$pk \leftarrow (pk', \mathsf{KDF}, \mathsf{MAC})$; Send pk to A
$(K^*, \kappa^*) \leftarrow \mathsf{KDF}(\tilde{K}_b^*)$; $\sigma^* \leftarrow \mathsf{MAC}(\kappa^*, \theta^*)$
$\phi^* \leftarrow (\theta^*, \sigma^*)$; Send (ϕ^*, K^*) to $\mathcal{A}$
If $\mathcal{A}$ submits (θ_i, σ_i) to the decapsulation oracle
  submit $(\theta_i, \mathrm{pred}_i)$ to its CDecap oracle
  to get $\tilde{K}_i$ or $\perp$. Simulate the decapsulation oracle
  of $\mathcal{A}$ if $\tilde{K}_i$ is returned or send $\perp$ to $\mathcal{A}$
If $\mathcal{A}$ outputs β', output it as b'

Note that the above $\mathcal{C}$ perfectly simulates the environment of $\mathcal{A}$ in Game $\mathsf{G_3}$. Consequently, we have

$$|\Pr[S_3] - \frac{1}{2}| \leq \mathbf{Adv}^{\text{IND-CCCA}}_{\mathcal{C},\mathsf{KEM'}}(\lambda).$$

It remains to show that the attackers $\mathcal{C}$'s against the IND-CCCA security of the $\mathsf{KEM'}$ are admissible, i.e. the corresponding plaintext uncertainty

$$\mathrm{uncert}_{\mathcal{C}}(\lambda) = \frac{1}{Q_{\mathcal{C}}} \sum_{1 \leq i \leq Q_{\mathcal{C}}} \Pr[\mathrm{pred}_i(\tilde{K}) = 1 \mid \tilde{K} \xleftarrow{\$} S_{\tilde{K}}]$$

is negligible for all PPT attackers $\mathcal{C}$'s constructed in the above games. We proceed to show the result for the attacker $\mathcal{C}$ in Game $\mathsf{G_2}$, the result for $\mathcal{C}$'s in Game $\mathsf{G_3}$ is obtained identically. To this end, we build an attacker $\mathcal{D}$ against the SUF-CMA security of the MAC scheme MAC. $\mathcal{D}$ gets as input the security parameter λ, the description of MAC and access to the MAC generation oracle $\mathsf{MAC.Sign}(\kappa^*, \cdot)$, where

κ^* is chosen uniformly at random from S_κ. $\mathcal{D}$ plays the role of the challenger for $\mathcal{C}$. $\mathcal{D}$ first runs $(pk', sk') \leftarrow \mathsf{KEM}'.\mathsf{Gen}(\lambda)$ and computes $(\theta^*, \tilde{K}) \leftarrow \mathsf{KEM}'.\mathsf{Encap}(pk')$. $\mathcal{D}$ then queries θ^* to its MAC-generation oracle to get $\sigma^* = \mathsf{MAC.Sign}(\kappa^*, \theta^*)$. $\mathcal{D}$ then gives (θ^*, σ^*) as a challenge ciphertext to $\mathcal{C}$. Since $\mathcal{D}$ knows the secret key sk', it can faithfully answer all of $\mathcal{C}$'s queries to the constrained decapsulation oracle. In the beginning, $\mathcal{D}$ picks a random index $j \leftarrow \{1, \ldots, Q_\mathcal{C}\}$, where $Q_\mathcal{C}$ is the number of queries to the constrained decapsulation oracle CDecap made by $\mathcal{C}$. On $\mathcal{C}$'s j-th decapsulation query $(\theta_j, \mathrm{pred}_{\sigma_j})$, $\mathcal{D}$ sets a fake predicate

$$\mathrm{pred}'_{\sigma_j}(\tilde{K}) := \begin{cases} 1 & : \quad \text{if } \mathsf{MAC.Ver}(\kappa^*, \sigma_j, \theta_j) = 1 \ \wedge \ \theta_j = \theta^* \ \wedge \\ & \quad\ \ \sigma_j \text{ was never returned by } \mathsf{MAC.Sign}(\kappa^*, \cdot) \\ 0 & : \quad \text{otherwise} \end{cases}$$

where $K||\kappa \leftarrow \mathsf{KDF}(\tilde{K})$ and outputs K if $\mathrm{pred}'_{\sigma_j}(\tilde{K}) = 1$. Note here that $\mathrm{pred}'_{\sigma_j}(\tilde{K})$ is defined in terms of $\kappa^* \xleftarrow{\$} S_\kappa$, while for the real predicate $\mathrm{pred}_{\sigma_j}(K)$, $(\nu, \kappa^*) \leftarrow \mathsf{KDF}(\tilde{K})$ with $\tilde{K}$ being chosen uniformly at random from $S_{\tilde{K}}$. Now let F_4 denote the event that $\mathrm{pred}_{\sigma_j}(\tilde{K}) \neq \mathrm{pred}'_{\sigma_j}(\tilde{K})$. Then,

$$\left|\Pr[\mathrm{pred}_{\sigma_j}(\tilde{K}) = 1 \mid \tilde{K} \xleftarrow{\$} S_{\tilde{K}}] - \Pr[\mathrm{pred}'_{\sigma_j}(\tilde{K}) = 1 \mid \tilde{K} \xleftarrow{\$} S_{\tilde{K}}]\right| \leq \Pr[F_4]$$

It is easy to see that $\Pr[F_4] \leq \mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda)$, for a suitable adversary $\mathcal{B}$. Therefore,

$$\begin{aligned} \mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda) &= \Pr[\mathsf{MAC.Ver}(\kappa^*, \sigma_j, \theta^*) = 1 \wedge \sigma_j \text{ was never returned by} \\ &\qquad \mathsf{MAC.Sign}(\kappa^*, \cdot)] \\ &= \frac{1}{Q_\mathcal{C}} \sum_{1 \leq j \leq Q_\mathcal{C}} \Pr[\mathrm{pred}'_j(\tilde{K}) = 1] \geq \\ &\geq \frac{1}{Q_\mathcal{C}} \sum_{1 \leq j \leq Q_\mathcal{C}} \left(\Pr[\mathrm{pred}_j(\tilde{K}) = 1] - \mathbf{Adv}^{\mathrm{ROR}}_{B_4,\mathsf{KDF}}(\lambda)\right) = \\ &= \mathrm{uncert}_\mathcal{C}(\lambda) - \mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda) \end{aligned}$$

and thus $\mathrm{uncert}_\mathcal{C}(\lambda) \leq \mathbf{Adv}^{\mathrm{ROR}}_{\mathcal{B},\mathsf{KDF}}(\lambda) + \mathbf{Adv}^{\text{SUF-CMA}}_{\mathcal{D},\mathsf{MAC}}(\lambda)$.

B Decisional Diffie-Hellman (DDH) Problem

We review the definition of the Decisional Diffie-Hellman (DDH) problem. Let $\mathcal{A}$ be an attacker. Let $\mathbb{G}$ be a finite cyclic group generated by $g \in \mathbb{G}$. Let p be a prime order of $\mathbb{G}$, whose size depends on the security parameter λ. We define the DDH problem using the attacker $\mathcal{A}$'s advantage in distinguishing two distributions:

$$\begin{aligned} \mathbf{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathbb{G}}(\lambda) = |\Pr[a \xleftarrow{\mathrm{R}} \mathbb{Z}_p; b \xleftarrow{\mathrm{R}} \mathbb{Z}_p : 1 \leftarrow \mathcal{A}(1^\lambda, g^a, g^b, g^{ab})] \\ - \Pr[a \xleftarrow{\mathrm{R}} \mathbb{Z}_p; b \xleftarrow{\mathrm{R}} \mathbb{Z}_p; r \xleftarrow{\mathrm{R}} \mathbb{Z}_p : 1 \leftarrow \mathcal{A}(1^\lambda, g^a, g^b, g^r)]|. \end{aligned}$$

We say that the DDH problem is hard if $\mathbf{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\lambda) = \max_\mathcal{A}\left\{\mathbf{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathbb{G}}(\lambda)\right\}$ is negligible for any attacker $\mathcal{A}$.

New Anonymity Notions for Identity-Based Encryption

Malika Izabachène and David Pointcheval

Ecole Normale Supérieure – LIENS/CNRS/INRIA, France
{Malika.Izabachene,David.Pointcheval}@ens.fr

Abstract. Identity-based encryption is a very convenient tool to avoid key management. Recipient-privacy is also a major concern nowadays. To combine both, anonymous identity-based encryption has been proposed. This paper extends this notion to stronger adversaries (the authority itself). We discuss this new notion, together with a new kind of non-malleability with respect to the identity, for several existing schemes. Interestingly enough, such a new anonymity property has an independent application to password-authenticated key exchange. We thus come up with a new generic framework for password-authenticated key exchange, and a concrete construction based on pairings.

1 Introduction

Motivation. The idea of using identities instead of public keys in order to avoid the (costly) use of certificates comes from Shamir [20]. He indeed suggested *Identity-based Encryption* ($\mathcal{IBE}$), that would allow a user to encrypt a message using any string, that would specify the recipient, as encryption parameter, such that this recipient only can decrypt the ciphertext.

Identity-based cryptography thus provides this interesting feature that one does not need authenticated public keys. Key managament is made simpler. Note however that a drawback is an *authority* that is required to generate the private keys for the users, according to their identities. This authority thus has the ability to decrypt any ciphertext. Privacy cannot be achieved with respect to this authority. Nevertheless, privacy of the plaintext is not the unique goal in cryptography, with encryption schemes. Privacy of the recipient may also be a requirement. Such a *key-privacy* notion has already been defined in the public-key setting in [3]. It has more recently been extended to the identity-based setting in [1], under the notion of *anonymity*. However, the security model in this $\mathcal{IBE}$ setting still trusts the authority. Whereas trusting the authority is intrinsic for privacy of the plaintext, it is not for the privacy of the recipient: a stronger anonymity notion is possible, with respect to the authority, but is it achievable for practical $\mathcal{IBE}$?

For efficiency reasons, the use of *Key Encapsulation Mechanisms* $\mathcal{KEM}$ have been shown as a preferable approach [22]. It consists in generating an ephemeral key and an encrypted version of the latter. The ephemeral key is thereafter used with a *Data Encryption Method* $\mathcal{DEM}$ to encrypt the message. In such

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 375–391, 2008.

a context, we are interested in the *semantic security* of the ephemeral key, and the *anonymity* of the recipient. In the identity-based context, Bentahar *et al.* [7] defined *Identity-based Key Encapsulation Mechanisms* $\mathcal{IB}$-$\mathcal{KEM}$. An anonymity notion with respect to the authority would then be an interesting feature.

Interestingly enough, this notion of anonymity with respect to the authority might have side applications. One of them is *password-authenticated key exchange* [6]. Such a protocol allows two players to establish a private channel, using a short secret as a sole authentication means. The latter is thus subject to exhaustive search, but such a short secret is very convenient for human beings.

Related Work. The idea of *identity-based encryption* is due to Shamir [20], in 1984. The first goal was to simplify public key management. However, the first practical solutions appeared in 2001 only [10,15]. Thereafter, many schemes have been proposed, based on pairing, factoring and lattices. Since such schemes were dealing with encryption, the main security notion was the semantic security [17].

Even if recipient-anonymity had already been addressed for public-key encryption [3] in 2001, anonymity for $\mathcal{IBE}$ has been proposed recently by Abdalla *et al.* [1], but as a simple extension of the previous public-key setting definition. In 2006, Gentry [16] and Boyen and Waters [12] presented the first anonymous $\mathcal{IBE}$ schemes without random oracles.

Our contributions. As already noticed in [1], *anonymity* might have some side applications to searchable encryption. In this paper, we deal with anonymity for $\mathcal{IB}$-$\mathcal{KEM}$, even with respect to the authority, the so-called *Key Anonymity with respect to the Authority* and denoted KwrtA-Anonymity: we first provide a formal security model, and then we discuss this security notion with existing schemes. We also consider a new non-malleability notion for the identity, that we call *identity-based non-malleability*: if one encrypts a message (or a key) for user U, one has no idea about the value obtained by another user U', whatever the relation between U and U' (or the identities) is.

Thereafter, we show that these security notions can also have side applications to password-authenticated key exchange. Such a *KwrtA-anonymous* and *identity-based non-malleability* $\mathcal{IB}$-$\mathcal{KEM}$ scheme can indeed be plugged into a password-authenticated two-party key exchange protocol, in the same vein as the IPAKE construction [14] did with trapdoor hard-to-invert group isomorphisms. Our security result holds in a stronger security model than usual (with an adaptive selection of passive and active attacks, as in [19]), but the construction still assumes the random-oracle model [5], as in [14].

Eventually, we provide an $\mathcal{IB}$-$\mathcal{KEM}$, that is both *KwrtA-anonymous* and *identity-based non-malleable*, in addition to the full-identity semantic security, against chosen-plaintext adversaries. This thus leads to a new password-authenticated two-party key exchange protocol.

2 Anonymous Identity-Based Encryption

Anonymity for public-key encryption schemes has first been introduced by Bellare *et al.* [3], under the *key privacy* security notion, and has been extended to identity-based encryption by Abdalla *et al.* [1].

In these papers, anonymity meant that even if the adversary chooses a message and two identities (or two public keys), and the challenger encrypts the message with one of the identities (or keys), the adversary cannot guess which one has actually been involved in the computation. This notion is quite strong for public-key encryption, but not that strong in the identity-based setting since it does not capture anonymity with respect to the authority that knows the master secret key, and even chooses the public parameters PK.

Unfortunately, the previous definitions cannot be trivially extended: the adversary can easily break anonymity if he knows the expected plaintext, and just hesitates between two identities, since he can decrypt any ciphertext. Anonymity can only be expected against the server if the plaintexts follow a non-trivial distribution. Since we will deal with key-encapsulation mechanisms, this non-trivial distribution is already implicit for the ephemeral keys.

This enhanced security notion will be called *Key Anonymity with respect to the Authority* and denoted $\mathsf{KwrtA\text{-}Anonymity}$. This section defines precisely this notion for identity-based key encapsulation mechanisms.

2.1 Identity-Based Encryption and Key Encapsulation Mechanisms

We first review the definitions of identity-based encryption, and more specifically of identity-based key encapsulation mechanisms [7]. In the following, we assume that identities are bit strings in a dictionary Dic.

Definition 1 (Identity-Based Encryption). *An $\mathcal{IBE}$ scheme is specified by four algorithms:*

$\mathsf{Setup}_{\mathsf{IBE}}(1^\lambda)$**.** *Takes as input a security parameter λ. It outputs the public parameters PK, as well as a master secret key MK.*

$\mathsf{Extract}_{\mathsf{IBE}}(\mathsf{MK}, \mathsf{ID})$**.** *Takes as input the master secret key MK, and the identity ID of the user. It outputs the user's decryption key usk.*

$\mathsf{Encrypt}_{\mathsf{IBE}}(\mathsf{PK}, \mathsf{ID}, M)$**.** *Takes as input the public parameter PK, the identity of the recipient, and a message M to be encrypted. It outputs a ciphertext.*

$\mathsf{Decrypt}_{\mathsf{IBE}}(\mathsf{usk}, c)$**.** *Takes as input the user's decryption key and a ciphertext c. It outputs the decryption or $\perp$, if the ciphertext is not valid.*

In [21], Shoup proposed a more efficient framework for public-key encryption, the so-called KEM/DEM, for *key encapsulation mechanism/data encapsulation method*. More recently, Bentahar *et al.* [7] extended this concept to the identity-based setting, and therefore proposed some constructions of $\mathcal{IB}$-$\mathcal{KEM}$ semantically secure. We will use the following formalism:

Definition 2 (Identity-Based Key Encapsulation Mechanism) *An $\mathcal{IB}$-$\mathcal{KEM}$ scheme is specified by the following four algorithms:*

$\mathsf{Setup}_{\mathsf{IBK}}(1^\lambda)$**.** *Takes as input a security parameter λ. It outputs the public parameters* PK*, as well as a master secret key* MK*.*

$\mathsf{Extract}_{\mathsf{IBK}}(\mathsf{MK}, \mathsf{ID})$**.** *Takes as input the master secret key* MK *and an identity* ID *of the user. It outputs the user's decryption key* usk*.*

$\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID})$**.** *Takes as input the public parameters* PK *and the identity of the recipient. It outputs a pair (K, c), where K is the ephemeral session key and c is the encapsulation of that key.*

$\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}, c)$**.** *Takes as input the user's decryption key* usk *and a ciphertext c. It outputs the key K encapsulated in c or $\perp$, if the ciphertext is not valid. We also formally define the function $\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}, c)$, which takes as input a user identity and a ciphertext c. It first extracts the decryption key* usk *associated to* ID*, and then decapsulates c under* usk*.*

We first review the notion of semantic security for $\mathcal{IB}$-$\mathcal{KEM}$, then we deal with anonymity, and an additional security notion, that we call *identity-based non-malleability*.

2.2 Security Notions

We directly describe the security notions for identity-based key encapsulation mechanisms, but one can easily derive them for identity-based encryption.

Semantic Security. The semantic security formalizes the privacy of the key. The security game, in the strongest security model (*i.e.* chosen-ciphertext and full-identity attacks) is the following one:

Setup : The challenger runs the $\mathsf{Setup}_{\mathsf{IBK}}$ algorithm on input 1^λ to obtain the public parameters PK, and the master secret key MK. It publishes PK.

Find stage: The adversary $\mathcal{A}$ adaptively issues the following queries:

- Extract query on input an ID: The challenger runs the Extract algorithm on input $(\mathsf{MK}, \mathsf{ID})$, and provides the associated decryption key usk.
- Decaps query on input an ID and a ciphertext c: The challenger first extracts the decryption key for ID, and then decrypts the ciphertext c with this key. It outputs the resulting ephemeral key, or $\perp$.

$\mathcal{A}$ outputs a target identity ID^*, on which no Extract-query has been asked.

Challenge: The challenger randomly gets $(K_0, c^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}^*)$ and $(K_1, c') \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}^*)$. It flips a bit b and outputs (K_b, c^*).

Guess stage: The adversary can issue the same queries as in the Find stage, with the restriction that no Extract-query on input ID^* and no Decaps-query on input (ID^*, c^*) can be asked. The adversary finally outputs its guess $b' \in \{0, 1\}$ for b.

We then define the advantage of $\mathcal{A}$ in breaking the *Semantic Security* of an $\mathcal{IB}$-$\mathcal{KEM}$ scheme with its ability in deciding whether it actually received the real ephemeral key associated to c or a random one. We denote this security

notion by IND, which can thereafter be combined with various oracle accesses, in order to define selective/full-identity and chosen plaintext/ciphertext attacks. More formally, we want the advantage below, to be negligible:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathsf{IBK}}(\mathcal{A}) = 2 \times \Pr_b \left[\begin{array}{c} (\mathsf{PK}, \mathsf{MK}) \leftarrow \mathsf{Setup}_{\mathsf{IBK}}(1^\lambda); (\mathsf{ID}^*, s) \leftarrow \mathcal{A}_1(\mathsf{PK}) \\ (K_0, c_0) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}^*); \\ (K_1, c_1) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}^*) \\ b' \leftarrow \mathcal{A}_2(K_b, c_0, s) : b = b' \end{array} \right] - 1.$$

In the following, we will need a very weak notion, that we call *weak semantic security*, during which attack that adversary has to choose in advance the target identity ID^* (selective-ID), and has no oracle access at all: no Decaps queries, and no Extract queries.

Anonymity. Anonymity against $\mathcal{IBE}$ means that for a chosen plaintext, and given a ciphertext c encrypted under ID_0 or ID_1 of adversary's choice, the adversary should not be able to decide which identity has been involved. With an appropriate $\mathcal{DEM}$ encryption scheme, the key encapsulation anonymity version can be defined as follows:

Setup: The challenger runs $\mathsf{Setup}_{\mathsf{IBK}}$ on input 1^λ to obtain the public parameters PK, and the master secret key MK. It publishes PK.

Find stage: The adversary $\mathcal{A}$ adaptively issues Extract and Decaps queries. $\mathcal{A}$ outputs two identities ID_0, ID_1, on which no Extract-query has been asked before.

Challenge: The challenger randomly selects $b \in \{0, 1\}$ and gets an encapsulated pair (K^*, c^*) under ID_b. It returns (K^*, c^*).

Guess stage: The adversary can issue the same queries as in the Find stage, subject to the restriction that no Extract-query is allowed to be asked on ID_0 or ID_1, and no Decaps-query can be asked on input (ID_0, c^*), or (ID_1, c^*). It finally outputs its guess $b' \in \{0, 1\}$ for b.

We say that an $\mathcal{IB}$-$\mathcal{KEM}$ scheme provides key-anonymity if the advantage of $\mathcal{A}$ in deciding which identity is actually involved in the above experiment is negligible:

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{IBK}}(\mathcal{A}) = 2 \times \Pr_b \left[\begin{array}{c} (\mathsf{PK}, \mathsf{MK}) \leftarrow \mathsf{Setup}_{\mathsf{IBK}}(1^\lambda); \\ (\mathsf{ID}_0, \mathsf{ID}_1, s) \leftarrow \mathcal{A}_1(\mathsf{PK}) \\ (K^*, c^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}_b); \\ b' \leftarrow \mathcal{A}_2(K^*, c^*, s) : b = b' \end{array} \right] - 1.$$

As already noticed, this anonymity notion does not provide any security with respect to the authority, since the above security notion assumes that the adversary has no idea about MK.

KwrtA-Anonymity. We therefore enhance the previous security model, in order to consider the authority as a possible adversary. However, it is clear that given (K^*, c^*), the authority can check the involved ID. We thus truncate the input to c^* only:

Find stage: The adversary generates (valid, see below) public parameters PK. $\mathcal{A}$ outputs PK and two identities ID_0, ID_1.

Challenge : The challenger randomly selects $b \in \{0,1\}$, and generates a ciphertext for ID_b, $(K^*, c^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}_b)$. It outputs c^*.

Guess stage: The adversary finally outputs its guess $b' \in \{0,1\}$.

We say that an $\mathcal{IB}$-$\mathcal{KEM}$ scheme provides *Key Anonymity with respect to the Authority* (denoted $\mathsf{KwrtA\text{-}Anonymity}$) if the advantage of $\mathcal{A}$ in deciding which identity is involved in the experiment above is negligible:

$$\mathsf{Adv}^{\mathsf{kwrta-anon}}_{\mathsf{IBK}}(\mathcal{A}) = 2 \times \Pr_b \left[\begin{array}{c} (\mathsf{PK}, \mathsf{ID}_0, \mathsf{ID}_1, s) \leftarrow \mathcal{A}_1(1^\lambda) \text{ s.t. } \mathsf{Valid}_{\mathsf{IBK}}(\mathsf{PK}) \\ (K^*, c^*) \leftarrow \mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID}_b); \\ b' \leftarrow \mathcal{A}_2(c^*, s) : b = b' \end{array} \right] - 1.$$

We emphasis that in the above experiment, the adversary has to generate valid public parameters PK. Note that against $\mathsf{KwrtA\text{-}Anonymity}$ (*vs.* anonymity), on the one hand, the new adversary may know the master key MK, but on the other hand, it must make its decision from c^* only. Therefore, these two security notions are not really comparable. Furthermore, since the adversary generates PK, one has to be able to check the honest generation. In some cases, PK is a truly random value, without redundancy; in some other cases, appropriate redundancy should be proven. We thus define an additional algorithm:

$\mathsf{Valid}_{\mathsf{IBK}}(\mathsf{PK})$**.** Takes as input the public parameters PK, and checks whether they satisfy the required properties.

Identity-based Non-Malleability. In the application we will study later, a new security notion for identity-based encryption will appear. It basically states that when one sends a ciphertext to a user ID, one has no idea how user ID' will decrypt it, even for identities chosen by the adversary. This means that when one computes an encapsulation, it provides an ephemeral session key with a unique recipient, and not several secret keys with several partners. We define the *identity-based non-malleability* game as follows:

Setup: The challenger runs $\mathsf{Setup}_{\mathsf{IBK}}$ on input 1^λ to obtain the public parameters PK, and the master secret key MK. It publishes PK.

Attack: The adversary $\mathcal{A}$ adaptively issues $\mathsf{Extract}$ and Decaps queries, and outputs a ciphertext c, and two pairs (K_0, ID_0), and (K_1, ID_1).

The adversary wins this game if the two formal equalities hold:

$$K_0 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_0, c) \text{ and } K_1 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_1, c).$$

We thus define the success of $\mathcal{A}$ in breaking the Identity-based Non-Malleability of an $\mathcal{IB}$-$\mathcal{KEM}$ scheme by:

$$\mathsf{Succ}^{\mathsf{id\text{-}nm}}_{\mathsf{IBK}}(\mathcal{A}) = \Pr \left[\begin{array}{c} (\mathsf{PK}, \mathsf{MK}) \leftarrow \mathsf{Setup}_{\mathsf{IBK}}(1^\lambda); \\ (c, (K_0, \mathsf{ID}_0), (K_1, \mathsf{ID}_1)) \leftarrow \mathcal{A}(\mathsf{PK}) : \\ K_0 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_0, c) \wedge K_1 = \mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{ID}_1, c) \end{array} \right].$$

Note that this security notion is for a normal user, and not for the authority itself. Indeed, it would clearly be incompatible with $\mathsf{KwrtA\text{-}Anonymity}$.

3 Anonymous and Non-malleable $\mathcal{IB}$-$\mathcal{KEM}$

Since the first practical $\mathcal{IBE}$ schemes, new features, and new efficient/security criteria have been defined. An efficient anonymous $\mathcal{IBE}$ with a tight security proof in the standard model is one of the open problems. In this section, we first review some candidates, and then propose a new scheme that satisfies all the above requirements: semantic security, various anonymity notions and identity-based non-malleability.

3.1 Backgrounds on Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of large prime order p. We suppose that these two groups are equipped with a pairing, *i.e.* a non-degenerated and efficiently computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In the following, we use multiplicative notation for $\mathbb{G}_1$ and $\mathbb{G}_2$: $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$, for all $a, b \in \mathbb{Z}_p$, and any $g_1 \in \mathbb{G}_1$ and $g \in \mathbb{G}_2$.

For the sake of generality, we consider the asymmetric case, where $\mathbb{G}_1 \neq \mathbb{G}_2$, but most of the schemes below also apply in the symmetric setting, where $\mathbb{G}_1 = \mathbb{G}_2$.

3.2 Diffie-Hellman Assumptions

The co-CDH-Problem. Let g_1 and g_2 two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We define the co-Diffie-Hellman value $\mathsf{co\text{-}CDH}_{g_1,g_2}(u)$, for $u = g_1^x \in \mathbb{G}_1$, the element $v = g_2^x \in \mathbb{G}_2$.

The $\mathsf{co\text{-}CDH}_{\mathbb{G}_1,\mathbb{G}_2}$ problem can be formalized as follows: given $g_1, u \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, output $v = \mathsf{co\text{-}CDH}_{g_1,g_2}(u)$. We define the success probability of $\mathcal{A}$ in breaking the $\mathsf{co\text{-}CDH}_{\mathbb{G}_1,\mathbb{G}_2}$-problem as:

$$\mathsf{Succ}^{\mathsf{co-cdh}}_{\mathbb{G}_1,\mathbb{G}_2}(\mathcal{A}) = \Pr\left[g_1 \stackrel{R}{\leftarrow} \mathbb{G}_1; g_2 \stackrel{R}{\leftarrow} \mathbb{G}_2, x \stackrel{R}{\leftarrow} \mathbb{Z}_p; v \leftarrow \mathcal{A}(g_1, g_2, g_1^x) : v = g_2^x\right].$$

Note that when $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, the $\mathsf{co\text{-}CDH}_{\mathbb{G},\mathbb{G}}$-problem is exactly the usual *Computational Diffie-Hellman Problem* in $\mathbb{G}$, which can still be difficult. However, the decisional version is easy, granted the pairing.

We can indeed define the $\mathsf{co\text{-}DH}_{\mathbb{G}_1,\mathbb{G}_2}$-language of the quadruples $(a, b, c, d) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_2$, such that $d = \mathsf{co\text{-}CDH}_{a,b}(c)$.

The Common co-CDH-Problem. Given two elements, it is simple to complete a co-CDH-quadruple (g_1, g_2, u, v). However, finding two such quadruples with constraints may not be simple. We thus define a new problem, called the *Common* co-CDH-*Problem*, as follows: Given $g, h \in \mathbb{G}$, and $V \in \mathbb{G}_T$, output $k_0 \neq k_1 \in \mathbb{Z}_p$, $K_0, K_1 \in \mathbb{G}_T$ and a common $c \in \mathbb{G}$, such that:

$$(gh^{k_0}, V, c, K_0), (gh^{k_1}, V, c, K_1) \in \mathsf{co\text{-}DH}_{\mathbb{G},\mathbb{G}_T}.$$

We define the success of $\mathcal{A}$ in breaking the $\mathsf{Common\text{-}co\text{-}CDH}_{\mathbb{G},\hat{e}}$-*Problem* as:

$$\mathsf{Succ}^{\mathsf{common\text{-}co\text{-}cdh}}_{\mathbb{G},\hat{e}}(\mathcal{A}) = \Pr\left[\begin{array}{l} g,h \in \mathbb{G}; V \in \mathbb{G}_T; (c,k_0,k_1,K_0,K_1) \leftarrow \mathcal{A}(g,h,V): \\ \quad k_0 \neq k_1 \wedge (gh^{k_0},V,c,K_0) \in \mathsf{co\text{-}DH}_{\mathbb{G},\mathbb{G}_T} \\ \quad \wedge (gh^{k_1},V,c,K_1) \in \mathsf{co\text{-}DH}_{\mathbb{G},\mathbb{G}_T} \end{array}\right]$$

The CBDH-Problem. Diffie-Hellman variants have been proposed in groups equipped with pairings, and namely in the symmetric case: let g be a generator of $\mathbb{G}$. We define the Bilinear Diffie-Hellman value of g^x, g^y, g^z, for $x,y,z \in \mathbb{Z}_p$, in base g, the element $V = \hat{e}(g,g)^{xyz} \in \mathbb{G}_T$.

The $\mathsf{CBDH}_{\mathbb{G},\hat{e}}$ problem can be formalized as follows: given $g, X = g^x, Y = g^y, Z = g^z \in \mathbb{G}$, output $V = \hat{e}(g,g)^{xyz}$. We define the success probability of $\mathcal{A}$ in breaking the $\mathsf{CBDH}_{\mathbb{G},\hat{e}}$-problem as:

$$\mathsf{Succ}^{\mathsf{cbdh}}_{\mathbb{G},\hat{e}}(\mathcal{A}) = \Pr\left[g \xleftarrow{R} \mathbb{G}; x,y,z \xleftarrow{R} \mathbb{Z}_p; V \leftarrow \mathcal{A}(g,g^x,g^y,g^z) : v = \hat{e}(g,g)^{xyz}\right].$$

The DBDH-Problem. The decisional version can then be intractable too: given $g, X = g^x, Y = g^y, Z = g^z \in \mathbb{G}$, and $V \in \mathbb{G}_T$, decide whether $V = \hat{e}(g,g)^{xyz}$, or not. We define the advantage of $\mathcal{A}$ in breaking the $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$-problem as:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{dbdh}}_{\mathbb{G},\hat{e}}(\mathcal{A}) = {} & \Pr\left[g \xleftarrow{R} \mathbb{G}; x,y,z \xleftarrow{R} \mathbb{Z}_p; V = \hat{e}(g,g)^{xyz} : 1 \leftarrow \mathcal{A}(g,g^x,g^y,g^z,V)\right] \\ & - \Pr\left[g \xleftarrow{R} \mathbb{G}; x,y,z \xleftarrow{R} \mathbb{Z}_p; V \xleftarrow{R} \mathbb{G}_T : 1 \leftarrow \mathcal{A}(g,g^x,g^y,g^z,V)\right].\end{aligned}$$

The Successive-Power Version. For our scheme to be semantically secure, we will need a stronger variant of the above DBDH problem, given access to a sequence of powers, similarly to the Strong Diffie-Hellman problem [9]: More precisely, given g, g^x, g^y, g^z, and $g^{z/x}, g^{z/x^2}, \ldots, g^{z/x^q}$, as well as V, from some $V \in \mathbb{G}_T$, where q is a parameter, decide whether $V = \hat{e}(g,g)^{xyz}$, or a random element. We define the advantage of $\mathcal{A}$ in breaking the q-$\mathsf{SP\text{-}DBDH}_{\mathbb{G},\hat{e}}$-assumption as:

$$\begin{aligned}\mathsf{Adv}^{q\text{-}\mathsf{spdbdh}}_{\mathbb{G},\hat{e}}(\mathcal{A}) = {} & \Pr\left[\begin{array}{l} g \xleftarrow{R} \mathbb{G}; x,y,z \xleftarrow{R} \mathbb{Z}_p; V = \hat{e}(g,g)^{xyz} : \\ \quad 1 \leftarrow \mathcal{A}(g,g^x,g^y,g^z,g^{z/x},\cdots,g^{z/x^q},V) \end{array}\right] \\ & - \Pr\left[\begin{array}{l} g \xleftarrow{R} \mathbb{G}; x,y,z \xleftarrow{R} \mathbb{Z}_p; V \xleftarrow{R} \mathbb{G}_T : \\ \quad 1 \leftarrow \mathcal{A}(g,g^x,g^y,g^z,g^{z/x},\cdots,g^{z/x^q},V) \end{array}\right].\end{aligned}$$

It is clear that such a sequence of powers should not provide much information to the adversary. And thus, for any polynomial-time adversary $\mathcal{A}$, the above advantage is negligible. In the full version of this paper [18], we provide the proofs that our two new problems are intractable for generic adversaries.

3.3 Previous $\mathcal{IBE}$ Schemes

Let us review several $\mathcal{IBE}$, and see which properties they satisfy. For the sake of simplicity, for all of them, we review the key encapsulation mechanisms. In

several schemes, we will need a deterministic map F from identities onto the group $\mathbb{G}$, possibly with parameter PK.

The Boneh-Franklin Scheme [10]. In this scheme, $\mathsf{MK} = s \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $\mathsf{PK} = g^s$. The map $F(\mathsf{ID})$ is independent of PK. This is a function onto $\mathbb{G}$, modeled as a random oracle in the security analysis. The ciphertext $c = g^r \in \mathbb{G}$ corresponds to the key $K = \hat{e}(F(\mathsf{ID}), \mathsf{PK})^r = \mathsf{BDH}_g(\mathsf{PK}, c, F(\mathsf{ID})) = \hat{e}(\mathsf{usk}_{\mathsf{ID}}, c)$, where $\mathsf{usk}_{\mathsf{ID}} = F(\mathsf{ID})^s = \mathsf{co\text{-}CDH}_{g,F(\mathsf{ID})}(\mathsf{PK}) \in \mathbb{G}$.

It is quite similar to the ElGamal encryption, and thus the *semantic security* relies on the $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$, but against chosen-plaintext attacks only, in the random oracle model, even with access to the $\mathsf{Extract}$-query, which is similar to the Boneh-Lynn-Shacham signature [11] (secure against chosen-message attacks under the $\mathsf{CDH}_{\mathbb{G}}$ problem).

Since the ciphertext is totally independent of the identity, this scheme is *KwrtA-anonymous*, in the information-theoretical sense. Nevertheless, the basic anonymity is similar to the semantic security, and relies on the $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$. However, since the ciphertext does not involve the identity, it is easy to break the *identity-based non-malleability*: knowing r and $c = g^r$, one easily computes $K = \mathsf{BDH}_g(\mathsf{PK}, c, F(\mathsf{ID})) = \hat{e}(F(\mathsf{ID}), \mathsf{PK})^r$, for any ID of ones choice.

The Boneh-Boyen Scheme [8]. In this scheme, $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$, $g, g_2, h \stackrel{R}{\leftarrow} \mathbb{G}$, and $\mathsf{PK} = (g, g_1 = g^\alpha, g_2, h)$, while $\mathsf{MK} = g_2^\alpha$. The map F_{PK} is defined by $F_{\mathsf{PK}}(\mathsf{ID}) = g_1^{\mathsf{ID}} \cdot h$. The ciphertext $c = (g^s, F_{\mathsf{PK}}(\mathsf{ID})^s)$ corresponds to the key

$$K = \hat{e}(g_1, g_2)^s = \hat{e}(c_1, \mathsf{usk}_2)/\hat{e}(\mathsf{usk}_1, c_2),$$

if one gets $\mathsf{usk}_{\mathsf{ID}} = (g^r, \mathsf{MK} \cdot F_{\mathsf{PK}}(\mathsf{ID})^r)$, for any $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$.

As above, the *semantic security* relies on the $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$ assumption, in the standard model, but against selective-ID chosen-plaintext attacks, even with access to the $\mathsf{Extract}$-query (the underlying signature scheme is selective-forgery secure against chosen-message attacks under the CBDH assumption).

However, because of the redundancy in the ciphertext, which matches with one identity only, this scheme is not *anonymous*: one just has to check, for a candidate ID, and a ciphertext $c = (c_1, c_2)$, whether $(g, F_{\mathsf{PK}}(\mathsf{ID}), c_1, c_2)$ is a Diffie-Hellman tuple, by $\hat{e}(c_1, F_{\mathsf{PK}}(\mathsf{ID})) \stackrel{?}{=} \hat{e}(c_2, g)$. Since this attack did not need a candidate key K, a fortiori, this scheme is not *KwrtA-anonymous*.

On the other hand, since the ciphertext focuses to a specific recipient, one has no idea how another ID' would decrypt it, because of its randomness r' in the decryption key: for wrong user, with $\mathsf{usk}' = (g^{r'}, g_2^\alpha F_{\mathsf{PK}}(\mathsf{ID}')^{r'})$, and $c = (g^s, F_{\mathsf{PK}}(\mathsf{ID}')^{s'})$ ($s' \neq s$ since ID' is not the intended recipient), $K' = K \times H^{r'}$, for $H \neq 1$, and r' totally random. Therefore, it is *identity-based non-malleable* in the information-theoretical sense.

The Gentry Scheme [16]. In 2006, two schemes have been proposed, with provable anonymity. Gentry's scheme is one of them: $g, h \stackrel{R}{\leftarrow} \mathbb{G}$ and $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$. The public parameters are $\mathsf{PK} = (g, g_1 = g^\alpha, h)$ and $\mathsf{MK} = \alpha$. The map F_{PK} is defined

by $F_{\mathsf{PK}}(\mathsf{ID}) = g_1 \cdot g^{-\mathsf{ID}} = g^{\alpha-\mathsf{ID}}$. The ciphertext $c = (F_{\mathsf{PK}}(\mathsf{ID})^s, \hat{e}(g,g)^s)$ is the encapsulation of $K = \hat{e}(g,h)^s$, and thus, setting $(\mathsf{usk}_1, \mathsf{usk}_2) = (r, (hg^{-r})^{1/(\alpha-\mathsf{ID})})$, for any $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$, $K = \hat{e}(c_1, \mathsf{usk}_2) \cdot {c_2}^{\mathsf{usk}_1}$.

The scheme is *semantically secure* and *anonymous* against chosen plaintext attacks, even with access to the Extract-query, under the truncated decisional augmented bilinear Diffie-Hellman exponent assumption (see [16] for details).

However, the scheme is not *KwrtA-anonymous*, since using bilinear maps combined with the redundancy inside the ciphertext provides a test for any target identity ID', since knowing α, $\mathcal{A}$ can test whether

$$ {c_2}^{\alpha-\mathsf{ID}'} = e(g,g)^{s(\alpha-\mathsf{ID}')} \stackrel{?}{=} e(c_1, g) = e(g^{s(\alpha-\mathsf{ID}')}, g). $$

Since the ciphertext is specific to the recipient, $\mathcal{A}$ has no idea how an other ID' decrypts $c = (c_1, c_2)$, $c = (F_{\mathsf{PK}}(\mathsf{ID}')^{s'}, e(g,g)^s)$, since

$$ K' = \hat{e}(c_1, \mathsf{usk}'_2) \cdot {c_2}^{\mathsf{usk}'_1} = K \cdot (\hat{e}(g,g)^{\mathsf{usk}'_1} / \hat{e}(g,h))^{s-s'}, $$

is a random element in $\mathbb{G}_T$. Thus, the scheme is *identity-based non-malleable* in the information-theoretical sense.

The Boyen-Waters scheme [13]. Boyen and Waters proposed another provably anonymous scheme: ω, t_1, t_2, t_3 and $t_4 \stackrel{R}{\leftarrow} \mathbb{Z}_p$ are set to be the master secret key and $\Omega = \hat{e}(g,g)^{t_1 \cdot t_2 \cdot \omega}, g, g_0, g_1, v_1 = g^{t_1}, v_2 = g^{t_2}, v_3 = g^{t_3}$ are the public parameters PK, with g a random generator of $\mathbb{G}$ and $g_0, g_1 \stackrel{R}{\leftarrow} \mathbb{G}$. The map F_{PK} is defined by $F_{\mathsf{PK}}(\mathsf{ID}) = g_0 \cdot \mathsf{ID}$. To encrypt a key, one chooses a random $s \in \mathbb{Z}_p$ and sets $K = \Omega^s$, its encapsulation has the following form: $c = (c_0, c_1, c_2, c_3, c_4)$, with $c_0 = F_{\mathsf{PK}}(\mathsf{ID})^s$, $c_1 = v_1^{s-s_1}$, $c_2 = v_2^{s_1}$, $c_3 = v_3^{s-s_2}$, and $c_4 = v_4^{s_2}$. To decapsulate the key, one has to compute

$$ \begin{aligned} K^{-1} = \Omega^{-s} &= \hat{e}(g,g)^{-\omega t_1 t_2 s} \\ &= \hat{e}(c_0, \mathsf{usk}_0) \times \hat{e}(c_1, \mathsf{usk}_1) \times \hat{e}(c_2, \mathsf{usk}_2) \times \hat{e}(c_3, \mathsf{usk}_3) \times \hat{e}(c_4, \mathsf{usk}_4) \end{aligned} $$

with $\mathsf{usk}_{\mathsf{ID}} = (\mathsf{usk}_0, \mathsf{usk}_1, \mathsf{usk}_2, \mathsf{usk}_3, \mathsf{usk}_4)$, where:

$$ \begin{array}{ll} \mathsf{usk}_0 = g^{r_1 t_1 t_2 + r_2 t_3 t_4} & \\ \mathsf{usk}_1 = g^{-\omega t_2} F_{\mathsf{PK}}(\mathsf{ID})^{-r_1 t_2} & \mathsf{usk}_2 = g^{-\omega t_1} F_{\mathsf{PK}}(\mathsf{ID})^{-r_1 t_1} \\ \mathsf{usk}_3 = F_{\mathsf{PK}}(\mathsf{ID})^{-r_2 t_4} & \mathsf{usk}_4 = F_{\mathsf{PK}}(\mathsf{ID})^{-r_2 t_3} \end{array} $$

for any $r_1, r_2 \stackrel{R}{\leftarrow} \mathbb{Z}_p$. This scheme is *semantically secure* under $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$, and *anonymous* under the decision linear assumption (we do not give more details since this scheme is totally different from ours below. The reader is refereed to [13]). However, it is not *KwrtA-anonymous*: since knowing the master key and given a ciphertext $c = (c_0, c_1, c_2, c_3, c_4)$, one can decide for a target identity whether c_0, c_1, c_2 or/and c_0, c_3, c_4 is a linear tuple in basis v_0, v_1, v_2 and v_0, v_3, v_4 respectively.

Since the key is completely independent of the identity and c_0 is determined by the identity (among other elements), the same argument than for the two

previous schemes holds: it is *identity-based non-malleable* in an information-theoretically sense.

Note that for all the above schemes, the public parameters consist of independent elements in appropriate groups. The validity check $\mathsf{Valid}_{\mathsf{IBK}}(\mathsf{PK})$ is thus trivial.

3.4 Our Scheme

None of the previous schemes satisfies both *KwrtA-anonymity* and *identity-based non-malleability*. In this section, we describe our scheme, and show that it achieves all the security properties: *semantic security, anonymity, KwrtA-anonymity* and *identity-based non-malleability*. For the sake of simplicity, we use a symmetric pairing:

$\mathsf{Setup}_{\mathsf{IBK}}$**.** The setup algorithm chooses two random generators $g, h \in \mathbb{G}$, and a random exponent $\omega \in \mathbb{Z}_p$. It keeps this exponent as the master key $\mathsf{MK} = \omega$. The corresponding system parameters are: $\mathsf{PK} = (g, g_1 = g^\omega, h)$. It defines the identity-function: $F(\mathsf{ID}) = g_1 \cdot g^{\mathsf{ID}} = g^{\omega+\mathsf{ID}}$.
Note that, as above, the public parameters consist of independent elements in appropriate groups. The validity check $\mathsf{Valid}_{\mathsf{IBK}}(\mathsf{PK})$ is thus trivial.

$\mathsf{Extract}_{\mathsf{IBK}}(\mathsf{MK}, \mathsf{ID})$**.** To issue a private key for identity ID, the key extraction authority computes the private key, $\mathsf{usk}_{\mathsf{ID}} = h^{1/(\omega+\mathsf{ID})}$.

$\mathsf{Encaps}_{\mathsf{IBK}}(\mathsf{PK}, \mathsf{ID})$**.** In order to generate an ephemeral key with an identity ID, the algorithm chooses a random exponent $r \in \mathbb{Z}_p$, and creates the ciphertext as: $c = F(\mathsf{ID})^r$, that corresponds to the key $K = \hat{e}(g, h)^r$.

$\mathsf{Decaps}_{\mathsf{IBK}}(\mathsf{usk}_{\mathsf{ID}}, c)$**.** The decryption algorithm extracts the ephemeral key K from a ciphertext c by computing: $K = \hat{e}(\mathsf{usk}_{\mathsf{ID}}, c)$.

Correctness. Let us check the decryption process:

$$K = \hat{e}(\mathsf{usk}_{\mathsf{ID}}, c) = \hat{e}(h^{1/(\omega+\mathsf{ID})}, g^{r(\omega+\mathsf{ID})}) = \hat{e}(h, g)^r.$$

Semantic Security. It is worth to precise that we do not require to be able to simulate any oracle for making use of $\mathcal{IB}$-$\mathcal{KEM}$ schemes in the next section. The *weak semantic security* will be enough:

Theorem 3. *The weak semantic security of our scheme (under selective-ID, chosen-plaintext and no-identity attacks) relies on the* $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$*-problem, in the standard model.*

Proof. Given $u, A = u^a, B = u^b, C = u^c$, and $V \in \mathbb{G}_T$ the input to the $\mathsf{DBDH}_{\mathbb{G},\hat{e}}$-*Problem*, and the target identity ID^*, we set $g = A = u^a$, $h = C = u^c = g^{c/a}$, $g_1 = u^t \cdot A^{-\mathsf{ID}^*} = u^{t-a\mathsf{ID}^*}$, and $c = B$. This implicitly defines $\mathsf{MK} = t/a - \mathsf{ID}^*$, for a randomly chosen $t \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Therefore, $F_{\mathsf{PK}}(\mathsf{ID}^*) = g_1 g^{\mathsf{ID}^*} = u^t \cdot A^{-\mathsf{ID}^*} \cdot A^{\mathsf{ID}^*} = u^t$, and the randomness r of the challenge ciphertext $c = F_{\mathsf{PK}}(\mathsf{ID}^*)^r = u^{tr} = u^b = B$ is $r = b/t$. The corresponding encapsulated key should thus be

$$K = \hat{e}(h, g)^r = \hat{e}(u^c, u^a)^{b/t} = \hat{e}(u, u)^{abc/t}.$$

By letting $(V^{1/t}, c)$ be the output of the challenger, an adversary able to break the semantic security (without Extract-queries) helps us to decide whether V is the Bilinear Diffie-Hellman value or not. □

In order to show the usual semantic security (under full-ID, but chosen-plaintext attacks), we have to be able to simulate the Extract-oracle, which thus requires additional inputs. But first, we modify a little bit the scheme, by using $H(\mathsf{ID})$, instead of ID in the above description, where H is a random oracle [5] onto $\mathbb{Z}_p$.

Theorem 4. *The semantic security of our scheme (by using $H(\mathsf{ID})$, instead of ID) under full-ID and chosen-plaintext (no* Decaps *queries) relies on the* successive-power *version, in the random oracle model.*

Proof. Given u, $A = u^a$, $B = u^b$, $C = u^c$, $C_i = C^{1/a^i}$, for $i = 1, \ldots, q$, and $V \in \mathbb{G}_T$ the input to the q-SP-DBDH$_{\mathbb{G},\hat{e}}$-problem, we first compute $\{V_i = \hat{e}(u,u)^{bc/a^i}\}_{i=0\ldots q}$, since $V_0 = \hat{e}(B,C)$, and $V_i = \hat{e}(B,C_i)$, for $i = 1, \ldots, q$. Then, we set $g = A = u^a$ and $g_1 = u^t \cdot A^{-x^*}$, for randomly chosen $t, x^* \stackrel{R}{\leftarrow} \mathbb{Z}_p$. This implicitly defines $\mathsf{MK} = t/a - x^*$. We also choose random elements $x_1, \ldots, x_q \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, and set $P(X) = \prod(tX + x_i)$, a polynomial of degree q, where the number of random oracle queries is $q+1$. We then set $h = C^{P(1/a)} = g^{cP(1/a)}$, which can be easily computed granted C, C_1, …, C_q.

First, all the random oracle queries will be answered by an $x^* + x_i$, or x^* (for a unique randomly chosen query): we hope to assign x^* to $H(\mathsf{ID}^*)$, the target identity, which happens with probability $1/q$. Let us assume this situation:

- By definition, as above, $F_{\mathsf{PK}}(\mathsf{ID}^*) = g_1 g^{H(\mathsf{ID}^*)} = u^t \cdot A^{-x^*} \cdot A^{x^*} = u^t$;
- For all the other identities, $H(\mathsf{ID}_j) = x_j$, and then usk_j can be computed as

$$h^{1/(\mathsf{MK}+x^*+x_j)} = C^{P(1/a)/(\mathsf{MK}+x^*+x_j)} = C^{P(1/a)/(t/a+x_j)} = C^{P_j(1/a)},$$

  where P_j is a polynomial of degree $q-1$. Then usk_j can be easily computed granted C, C_1, …, C_{q-1}. Hence the simulation of the Extract-oracle.

As above, the challenge ciphertext is set $c = B = u^b = F_{\mathsf{PK}}(\mathsf{ID}^*)^r$ for $r = b/t$. The corresponding encapsulated key should thus be

$$K = \hat{e}(g,h)^r = \hat{e}(u^a, u^{cP(1/a)})^{b/t} = (\hat{e}(u,u)^{abc})^{P(1/a)/t}.$$

Let us expand $P(X) = \sum_{i=0}^{i=q} p_i X^i$, and then

$$K = \hat{e}(u,u)^{abc \cdot p_0/t} \times \prod_{i=1}^{i=q} \hat{e}(u,u)^{bc/a^{i-1} \cdot p_i/t} = \left(\hat{e}(u,u)^{abc}\right)^{p_0/t} \times \prod_{i=1}^{i=q} V_{i-1}^{p_i/t}.$$

If $V = \hat{e}(u,u)^{abc}$, the correct key is $V^{p_0/t} \times \prod_{i=1}^{i=q} V_{i-1}^{p_i/t}$. In the random case, the same computation leads to a totally random key (note that $p_0 = \prod x_i \neq 0 \bmod p$). Then, by letting $(V^{p_0/t} \times \prod_{i=1}^{i=q} V_{i-1}^{p_i/t}, c)$ be the output of the challenger, an adversary able to break the semantic security helps us to decide whether V is the Bilinear Diffie-Hellman value or not. We thus break the q-SP-DBDH$_{\mathbb{G},\hat{e}}$-problem. □

Anonymity. The usual anonymity notion relies on the same assumption as the semantic security. Since the ciphertext consists of $c = F(\mathsf{ID})^r$, a random element in $\mathbb{G}$, whatever the identity ID. It is thus clearly *KwrtA-anonymous*, in the information-theoretical sense.

Theorem 5. *Our scheme is unconditionally KwrtA-anonymous.*

Idendity-based Non-Malleability. Let us consider the ciphertext c, and its decryption with respect to ID_i for $i \in \{0, 1\}$. In the following, r_i is formally defined by $c = F(\mathsf{ID}_i)^{r_i}$, and $K_i = \hat{e}(g, h)^{r_i}$. Thus, the identity-based non-malleability relies on the intractability of finding $c, \{\mathsf{ID}_i, K_i\}$, with $\mathsf{ID}_0 \neq \mathsf{ID}_1$ such that $r_i = \log_{\hat{e}(g,h)}(K_i) = \log_{F(\mathsf{ID}_i)}(c)$. This thus leads to a solution of the *Common* co-CDH-*Problem*.

Theorem 6. *The identity-based non-malleability of our scheme relies on the* Common co-CDH-Problem *in groups* $\mathbb{G}$ *and* $\mathbb{G}_T$.

4 $\mathcal{IBK} - \mathcal{PAKE}$: Our Password-Authenticated Key Exchange Protocol

The previous sections focused on identity-based key encapsulation mechanisms, and new anonymity properties. We now show how a weakly semantically secure $\mathcal{IB}$-$\mathcal{KEM}$, that is both KwrtA-anonymous and identity-based non-malleable, can be used to build a password-authenticated key exchange.

4.1 Description of Our Scheme

Our new scheme is generic. It basically consists in generating the session key using this $\mathcal{IB}$-$\mathcal{KEM}$, under the common password as the identity, see Figure 1. The other party can easily recover the session key. Security notions for semantic security and perfect forward secrecy follow from the (weak) semantic security and anonymity properties of the $\mathcal{IB}$-$\mathcal{KEM}$ scheme.

4.2 Security Analysis

Communication Model. We assume to have a fixed set of protocol *participants*, and each of them can be either a client or a server. They are all allowed to participate to several different, possibly concurrent, executions of the key exchange protocol. We model this by allowing each participant an unlimited number of *instances* able to initiate or participate to an execution of the protocol.

In the password-based scenario, the two parties share a low-entropy secret pw which is drawn from a small dictionary Dic. In the following, we assume that the distribution is uniform. More complex distributions could be considered.

We use the security model introduced by Bellare *et al.* [4], improved by Abdalla *et al.* [2] to consider the Real-or-Random security notion instead of the

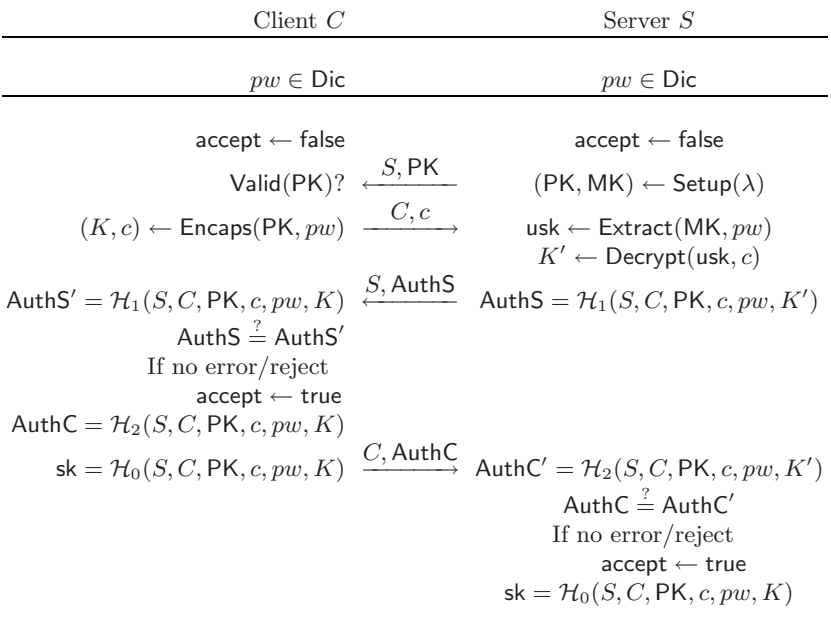


Fig. 1. IBK-PAKE: a Password-Authenticated Key-Exchange Protocol

Find-then-Guess. In this model, the adversary $\mathcal{A}$ has the entire control of the network, which is formalized by allowing $\mathcal{A}$ to ask the following query, $\mathsf{Send}(U, m)$, that models $\mathcal{A}$ sending the message m to instance U. The adversary $\mathcal{A}$ gets back the response U generates in processing the message m according to the protocol. A query $\mathsf{Send}(U, \mathsf{INIT})$ initializes the key exchange algorithm, by activating the first player in the protocol.

From the original security model, we suppress the Execute-queries. Even if they were important to model passive attacks *vs.* active attacks, we consider a stronger security model where the adversary always uses Send-queries, either for simply forwarding a flow generated by a honest user, or for modifying/manufacturing a flow. Thereafter, if the whole transcript of an execution of the protocol turns out to consist of forwarded flows only, this execution is then considered as a *passive* attack: it is similar to an Execute-query in previous models [4]. If one flow has been modified or manufactured, the session corresponds to an *active* attack.

As a consequence, in addition to the usual security model with Execute-queries, the adversary can adaptively decide, during an execution of the protocol, whether the session will correspond to a passive attack, or to an active one, and not from the beginning of the session only (as in [19]). An attack game will consist of a mix of passive and active attacks, in a concurrent manner.

However, as usual, we will be essentially interested in active attacks: q_{activeC} and q_{activeS} will, respectively, denote the number of active attacks in which the adversary played against the client and the server, respectively. We want to show

that $q_{\mathsf{activeC}}+q_{\mathsf{activeS}}$ is an upper-bound on the number of passwords the adversary may have tried.

Security Notions. Two main security notions have been defined for key exchange protocols. The first is the semantic security of the key, which means that the exchanged key is unknown to anybody other than the players. The second one is unilateral or mutual authentication, which means that either one, or both, of the participants actually know the key. In the following, we focus on the semantic security, also known as *AKE Security*.

The semantic security of the session key is modeled by an additional query $\mathsf{Test}(U)$. Since we are working in the Real-or-Random scenario, this Test-query can be asked as many times as the adversary $\mathcal{A}$ wants, but to *fresh* instances only. The freshness notion captures the intuitive fact that a session key is not "obviously" known to the adversary. More formally an instance is said to be fresh if it has successfully completed execution and

1. Neither it nor its partner was corrupted before the session started
2. or, the attack, on this session, was passive.

Two instances are partners if they run a key exchange protocol together. This is formally modeled by the notion of session ID: the session ID is a string defined from the transcript (usually, it consists of the first flows, sent and received), and two instances are partners if they share the same session IDs.

The Test-query is answered as follows: a (private) coin b has been flipped once for all at the beginning of the attack game, if $b = 1$ (Real), then the actual session key sk is sent back, if $b = 0$ (Random), or a random value is returned. Note that for consistency reasons, in the random case, the same random value is sent to partners.

We denote the AKE advantage as the probability that $\mathcal{A}$ correctly guesses the value of b with its output b': $\mathsf{Adv}^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$.

The adversary will also have access to the Corrupt-query that leaks the password: it is useful to model the perfect forward secrecy. The latter notion means that a session key remains secret even after the leakage of the long-term secret.

Security Result. For our protocol, we can state the following security result, which proof can be found in the full version [18].

Theorem 7 (AKE Security). *Let us consider an Identity-Based Key Encapsulation Mechanism* $\mathsf{IBK} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encaps}, \mathsf{Decaps})$ *that is weakly semantically secure (selective-ID, chosen-plaintext attacks and no* Extract*-queries),* KwrtA*-anonymous, and identity-based non-malleable, then our protocol* IBK-PAKE*, provides semantic security and perfect forward secrecy:*

$$Adv^{\mathsf{ake}}_{\mathsf{ibk}-\mathsf{pake}}(\mathcal{A}) \leq 4 \times \frac{q_{\mathsf{active}}}{N} + \mathsf{negl}(),$$

where $q_{\mathsf{active}} = q_{\mathsf{activeC}} + q_{\mathsf{activeS}}$ *is the number of active attacks and* N *is the size of the dictionary.*

5 Conclusion

In this paper, we have first introduced two new security notions for identity-based key encapsulation mechanisms: the first one is an enhancement of the usual anonymity, the second one formalizes a kind on non-malleability, with respect to the recipient identity.

Then, we proposed the first scheme that is full-ID semantically secure against chosen-message attacks, and that achieves our new security notions.

We furthermore showed that these new security notions could be useful for identity-based schemes as a tool: we provided a new framework for password-authenticated key exchange, with an identity-based key encapsulation mechanism as a core sub-routine.

Acknowledgment

We would like to thank the anonymous referees for their fruitful comments. This work has been partially supported by the French ANR PAMPA Project, and the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
3. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
5. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, pp. 62–73. ACM Press, New York (1993)
6. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
7. Bentaha, K., Farshim, P., Malone-Lee, J., Smart, N.P.: Generic constructions of identity-based and certificateless KEMs. Journal of Cryptology 21(2), 178–199 (2008)
8. Boneh, D., Boyen, X.: Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

9. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
10. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
11. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
12. Boneh, D., Waters, B.R.: Conjunctive, subset, and range queries on encrypted data. Cryptology ePrint Archive (2006)
13. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
14. Catalano, D., Pointcheval, D., Pornin, T.: IPAKE: Isomorphisms for password-based authenticated key exchange. Journal of Cryptology 20(1), 115–149 (2007)
15. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
16. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
17. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
18. Izabachène, M., Pointcheval, D.: New anonymity notions for identity-based encryption. In: SCN 2008. LNCS. Springer, Heidelberg (2008), `http://www.di.ens.fr/users/pointche`
19. Pointcheval, D., Zimmer, S.: Multi-factor authenticated key exchange. In: Bellovin, S.M., Gennaro, R., Keromytis, A., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 277–295. Springer, Heidelberg (2008)
20. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
21. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)
22. Shoup, V.: ISO 18033-2: An emerging standard for public-key encryption, Final Committee Draft (December 2004)

A Universally Composable Group Key Exchange Protocol with Minimum Communication Effort

Jun Furukawa[1], Frederik Armknecht[2], and Kaoru Kurosawa[3]

[1] NEC Corporation, Japan
j-furukawa@ay.jp.nec.com
[2] Ruhr-Universität, Germany
frederik.armknecht@trust.rub.de
[3] Ibaraki University, Japan
kurosawa@mx.ibaraki.ac.jp

Abstract. The universal composability (UC) framework by Canetti [15] is a general-purpose framework for designing secure protocols. It ensures the security of UC-secure protocols under arbitrary compositions. As key exchange protocols (KEs) belong to the most used cryptographic mechanisms, some research has been done on UC-secure 2-party KEs. However, the only result regarding UC-secure group key exchange protocols (GKEs) is a generic method presented by Katz and Shin [35]. It allows to turn any GKE protocol that fulfills certain security requirements into a UC-secure variant. This yields GKE protocols which require at least five communication rounds in practice when no session identities are provided by external mechanisms. Up to now, no effort has been taken to design dedicated UC-secure GKE protocols with a lower communication complexity.

In this paper, we propose a new UC-secure GKE which needs only two rounds. We show that two is the minimum possible number of rounds and that any 2-round UC-secure GKE requires at least as many messages as our protocol. The proof of security relies on a new assumption which is a combination of the decision bilinear Diffie-Hellman assumption and the linear Diffie-Hellman assumption.

Keywords: Group key exchange, universal composability, session ID generation.

1 Introduction

A preferable goal in provable security is to have security proofs which do not only show the security of protocols in an isolated environment but as well in composition with other protocol instances. Universally Composable (UC) security, introduced by Canetti in [15], provides this guarantee in a strong sense: A UC-secure protocol maintains its security properties even when composed concurrently with an unbounded number of instances of arbitrary protocols. Observe that UC supports a strong attacker model. Adversaries can control the whole communication, can control the schedules of all parties (e.g., delay messages) and can adaptively corrupt parties even during protocol runs. Not surprising, UC attracted a lot of attention and initiated a new research direction. For a variety of different types of protocols, e.g., signatures of different types (Canetti [16], Fischlin [28], Kurosawa et al. [37]), multi-party computation (Canetti et al. [23,22]),

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 392–408, 2008.

commitment schemes (Canetti et al. [18]), oblivious transfer (Fischlin [29]), mix-nets (Wikström [39]), etc., it has been examined if and how UC-secure instantiations can be designed. As key exchange protocols (KEs) belong to the most widely needed and used type of cryptographic protocols, the design of UC-secure KEs has been the topic of several works, e.g., Canetti et al. [21,19], Hofheinz et. al. [32], Le et al. [38].

However, the focus was mostly put on 2-party KEs whereas less is known about group key exchange protocols. The only result concerning UC-secure group key exchange protocols (GKEs) published so far was presented by Katz and Shin in [35]. Firstly, they gave a definition for the UC-security of GKEs. Capturing the right security notions for GKEs is a challenging task. Bresson et al. gave in [10,8,9] the first formal model of GKEs security, based on the formalization of two-party KEs given in [5,6,4] by Bresson et. al. They considered authenticated key exchange (AKE) (exchanged keys are indistinguishable from randomly chosen keys) and mutual authentication (MA) (each party is assured that its partners have generated the same key). The attacker model assumes that the participating parties are uncorrupted, but outsiders might control the communication (*outsider attacks*). Later on, several definitions to cover security in presence of malicious parties which participate to the GKE (*insider attacks*) have been presented [35,11,13,26].

The second result in [35] is a compiler that transforms any AKE-secure GKE into a UC-secure GKE. For example, the compiler could be applied to the modified version of the Burmester-Desmedt GKE [12] presented by Katz and Yung [36]. It is noteworthy that the Katz-Shin-compiler appends one extra communication round[1] to the underlying GKE in order to achieve UC-security which means an increase in the communication effort. To the best of our knowledge, all published AKE-secure GKEs have at least two rounds. Hence, according to current state of knowledge, the compiler can be used to generate UC-secure GKEs with a minimum number of three communication rounds. Note that this bound only holds within the UC framework where it is assumed that each protocol run is indexed by a unique session identifier (SID). This assumption is obviously not true in most practical environments so that the SID has to be generated during the protocol run. Barak et al. in [1] described a general protocol for this purpose , is to prepend two additional rounds in which nonces are exchanged. Summing up, generic methods are known to construct UC-secure GKEs which require five communication rounds.

However, for the sake of efficiency, the number of communication rounds and the number of exchanged messages should be as small as possible. Indeed, one can show (we give a proof in Appendix A) that any UC-secure GKE requires at least two rounds. Thus, the above mentioned results only prove the general feasibility of designing UC-secure GKEs but do not yield optimal results regarding the number of rounds and exchanged messages. Up to now, no dedicated UC-secure GKEs have been published which aim for a better communication effort. Observe that in this paper, we consider only group key exchange protocols where no pre-existing trust exists. In the case that one party is trusted by the others, this party could act as a kind of group leader who simply chooses the group key. This would make more efficient protocols are possible.

[1] Protocol runs are usually divided into communication rounds where each party sends within one round only messages which depend on its present state and the messages exchanged in previous rounds.

In this paper, we present the first UC-secure GKE which requires only two rounds, that is with the minimum number of rounds. Furthermore, we show that no two round UC-secure GKE can exist which requires less messages. We achieve this result by integrating the additional round from the Katz-Shin-compiler into the protocol and by generating the SID during the protocol. Each of these measurements saves one round but requires new concepts and dedicated security proofs.

The paper is organized as follows: Section 2 shortly explains the UC framework and introduces our security definitions for UC-secure GKE where the SIDs are not provided externally. Section 3 proposes our 2-round GKE while Section 4 proves its security within the UC framework. Section 5 concludes the paper.

2 Init-GKEs within the Universally Composable Framework

2.1 The Universally Composable Framework

We shortly recapitulate the basic ideas of the universally composable framework (UC framework) [15]. Informally, a protocol π realizes a cryptographic task in an UC-secure way if no pair of any environment $\mathcal{Z}$ and attacker $\mathcal{A}$, can distinguish the protocol's execution, called *real execution*, from the execution of an ideal functionality $\mathcal{F}$, called *ideal execution*. $\mathcal{F}$ can be seen as a kind of black box that ideally realizes the considered cryptographic task. For example in our case, this task would be to distribute keys between members of a group. The attacker can control the whole communication, can control the schedules of all parties, can adaptively corrupt any parties even during protocol runs, and can concurrently run arbitrary protocols for each party.

The real execution, denoted with $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$, is initiated by the environment $\mathcal{Z}$ on some input z and security parameter k. More precisely, $\mathcal{Z}$ activates an attacker $\mathcal{A}$ and multiple parties $\Pi_1,\ldots,\Pi_n$ running π. $\mathcal{A}$ is assumed to have total control over the content and the schedule of the communication between the parties $\{\Pi_i\}$. Furthermore, it can corrupt parties. At the end, $\mathcal{Z}$ outputs a value 0 or 1.

In contrast to the real execution exists the ideal execution $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ which differs in several points. The real parties $\Pi_1,\ldots,\Pi_n$ are "replaced" by dummy parties $\Phi_1,\ldots,\Phi_n$ which have the same input interface but interact with the ideal functionality $\mathcal{F}$ instead of running π. When a dummy party is invoked by a message it simply forwards it to $\mathcal{F}$ while a real party would start the protocol π. The output of a dummy party Φ is the output provided by $\mathcal{F}$ to Φ while the output of a real party would be its result from the execution of π. In a similar manner, the real adversary $\mathcal{A}$ is "replaced" by an ideal adversary $\mathcal{S}$. Again, if $\mathcal{Z}$ halts at some point in time, it outputs 0 or 1.

Now, π is defined to be UC-secure i.e., UC-realizes $\mathcal{F}$, if for any real adversary $\mathcal{A}$ exists an ideal adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can distinguish $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ from $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The notion of indistinguishability is formalized by the following definition:

Definition 1. *[15] A function f is* negligible *in a (security) parameter k (denoted by $f \approx 0$) if for any $c \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ such that for all $k > k_0$ we have $|f(k)| < k^{-c}$).*

Two executions $Exec_1$ and $Exec_2$ are indistinguishable (written $Exec_1 \approx Exec_2$) if

$$|Pr[Exec_1(k,z)=1] - |Pr[Exec_2(k,z)=1]| \approx 0.$$

2.2 An Ideal Functionality for SID-Generating Group Key Exchange Protocols

As already pointed out, one basic assumption within the UC framework is that protocol executions are labeled with unique session identifiers (SID). If these are not provided by some external mechanism, the participants in the protocol are required to generate them. This might be accomplished by an initialization protocol which, of course, has to be UC-secure as well. such as the one proposed in Barak et al. [1] described an appropriate ideal functionality $\mathcal{F}_{init}$ and proposed a two-round protocol which UC-realizes it. Thus, a naive approach would be to construct an UC-secure GKE which relies on the existence of SIDs and to prepend the Barak-et-al.-protocol. Obviously, it would be more efficient if we could integrate protocol initialization and group key exchange within one protocol (init-GKE). Therefore, we consider a functionality that realizes both tasks and consequently term it $\mathcal{F}_{init-GKE}$. Observe that the concept of init-GKEs has been considered before, e.g., [27,10,8,9,11], but never within the UC-framework.

Next, we introduce an ideal functionality $\mathcal{F}_{\text{init-gke}}$ for init-GKE protocols. According to the UC framework, whenever a party Φ wants to participate in a protocol execution, it invokes an instance (or copy) of itself and chooses a *party instance identifier piid* for it. Honest parties are expected to choose always a different party instance identifier for each party instance. In the following, we refer to a party by Φ and to one of its instances by $(\Phi, piid)$. While a party Φ might possess some long-term secrets, e.g., a signature key, party instances $(\Phi, piid)$ are invoked with an empty state. Because of this, a party instance reveals no long-term secret when it is corrupted. Long-term secrets are only revealed when parties (but not party instances) are corrupted. However $(\Phi, piid)$ might make use of Φ's long-term secret, e.g., by letting Φ sign the messages of $(\Phi, piid)$. This modeling is in compliance with the such model fits to the well accepted model of KEs [20].

Different protocol executions are referred to by *session identifiers sid.* As the generation of a *sid* will be part of the protocol, our functionality accepts requests from party instances which are not connected a-priori to some *sid* (in contrast to the GKE-functionality from [35]). For this purpose, we define the following sets which are managed by $\mathcal{F}$:

- READY(PID) is the set of all requests for participation in a group key exchange protocol where the group is specified by a set of party identifiers *PID*.
- CORR contains all sessions where at least one party is corrupted and no key has been generated yet
- KEYS is composed of all keys that have been generated so far

The ideal functionality $\mathcal{F}_{\text{init-gke}}$ is defined as in the following.

Definition 2. The ideal functionality $\mathcal{F}_{\text{init-gke}}$ *communicates with some party-instances and an ideal adversary $\mathcal{S}$. $\mathcal{F}_{\text{init-gke}}$ runs on a security parameter k and on a set $\mathcal{D}$ with $|\mathcal{D}| \in \mathcal{O}(2^k)$. $\mathcal{F}_{\text{init-gke}}$ stores the sets READY(PID), CORR, and KEYS as internal state. Here, a set READY(PID) is prepared for every (PID) when required.*

Participation Request: *Upon receiving a request* $(PID, (\Phi, piid), \mathsf{new\text{-}session})$ *from a party* Φ, $\mathcal{F}_{\text{init-gke}}$ *checks that* $\Phi \in PID$. *If (and only if) this is the case, it adds* $(\Phi, piid)$ *to the set* $READY(PID)$. *Afterwards,* $\mathcal{F}_{\text{init-gke}}$ *sends* $(PID, (\Phi, piid))$ *to* $\mathcal{S}$.

Key Generation: *Whenever* $\mathcal{F}_{\text{init-gke}}$ *receives a message* (PID, sid, ok) *from* $\mathcal{S}$, *it checks if* $sid = \{(\Phi_i, piid_i)\}_{i=1,\ldots,n}$, *and* $sid \subset READY(PID)$. *If this is the case,* $\mathcal{F}_{\text{init-gke}}$ *removes* sid *from* $READY(PID)$. *If this is not the case* $\mathcal{F}_{\text{init-gke}}$ *skips the rest of the process.*

If all party-instances $(\Phi_i, piid_i) \in sid$ *are uncorrupted,* $\mathcal{F}_{\text{init-gke}}$ *randomly chooses a key* $\kappa \in \mathcal{D}$ *and adds* (PID, sid, κ) *to KEYS. If any of the party-instances* $(\Phi_i, piid_i) \in sid$ *is corrupted,* $\mathcal{F}_{\text{init-gke}}$ *adds* (PID, sid) *to CORR but stores no entry in KEYS.*

Corrupted Key Generation: *Upon receiving a message* $(PID, sid, \mathsf{key}, \kappa)$ *from* $\mathcal{S}$, $\mathcal{F}_{\text{init-gke}}$ *checks if* $(PID, sid) \in CORR$. *If it is, it deletes this entry from CORR and adds* (PID, sid, κ) *to KEYS.*

Key Delivery: *If* $\mathcal{S}$ *sends a message* $(PID, sid, \mathsf{deliver}, (\Phi, piid))$ *where there is a recorded tuple* $(PID, sid, \kappa) \in KEYS$ *and* $(\Phi, piid) \in sid$, *then* $\mathcal{F}_{\text{init-gke}}$ *sends* $((\Phi, piid), PID, sid, \kappa)$ *to the party* Φ.

Party Instance Corruption: $\mathcal{S}$ *can request to corrupt any party instance* $(\Phi_i, piid_i) \in sid$ *which is marked as corrupted from this point on. In addition, if there is a recorded tuple* $(PID, sid, \kappa) \in KEYS$ *and a message* $((\Phi_i, piid_i), PID, sid, \kappa)$ *has not yet been sent to* Φ_i, *then* $\mathcal{S}$ *is given* κ, *otherwise nothing.*[2]

Observe that the functionality allows the parties to freely choose sets of possible group partners, hence giving them full flexibility.

As opposed to the definition given in [35], the keys are randomly chosen from a set $\mathcal{D}$ instead of $\{0,1\}^k$. This choice was made to make $\mathcal{F}_{\text{init-gke}}$ compliant to the proposed protocol π where keys come from a group $\tilde{\mathcal{G}}$. This yields no limitation of the model. Once a group of parties succeeds to share a key with a large enough entropy, one can smooth this entropy by applying a 2-universal hash function on it with a uniformly chosen public string. This does not require any further interaction between the parties and has therefore no impact on the round complexity. The leftover hash lemma [33,34] guarantees that the obtained key is indistinguishable from random string in $\{0,1\}^k$. For this purpose, the system should additionally provide a randomly chosen 2-universal hash function and a uniformly chosen public string. The same string can be used for every key exchange.

In [35], key exchange functionality is defined so as to be invoked for a single session. To capture the execution of multiple session, the multi-session extension of each functionality is considered. Then, according to the joint-state theorem [24], any GKE

[2] This is necessary to deal with the situation that the corrupted party instance is in the last round and the adversary withholds the final messages from this party instance. In this case, the adversary who learned the internal state (by corruption), can compute the key by himself while the corrupted party is unable to do so as it is still waiting for the final messages. On the other hand, if the corrupted party instance did already finish the key generation, the key is handed in the real world to the invoking party and the internal state is deleted. Hence, the adversary does learn nothing then.

protocol that UC-realizes the GKE functionality from [35] UC-realizes its multiple session extension as well. In $\mathcal{F}_{\text{init-gke}}$, whenever a party-instance (Φ, piid) is invoked, the session in which it will participate is not determined. Furthermore, each party may be invoked several times and the attacker has the full control over forming and scheduling sessions. Hence, our ideal functionality is intrinsically multi-session without the need for additional mechanisms.

Observe that for the case of a single execution with a fixed (possibly externally given) *sid*, $\mathcal{F}_{\text{init-gke}}$ behaves exactly like $\mathcal{F}_{GKE}$ from [35]. Hence, any combination of the initialization protocol from [1] and a protocol constructed with the Katz-Shin-compiler UC-realizes $\mathcal{F}_{\text{init-gke}}$. In the next Section, we start from such a construction and explain how the number of rounds can be reduced.

3 The Protocol

3.1 Problems and Our Approach

Before we describe our protocol in the next (sub-)section, we first sketch our approach and which challenges need to be overcome. In principle, we follow the ideas from the Katz-Shin compiler [35] and the initialization protocol from Barak et al. [1]. To explain these, let $\hat{\pi}$ be a GKE that UC-realizes $\mathcal{F}_{\text{init-gke}}$ and that has been compiled by applying the Katz-Shin compiler to an AKE-secure GKE and by prepending the initialization protocol from Barak et. al. For example, if one applies these modifications to the Burmester-Desmedt protocol [12] (in its AKE-secure variant given in [36]), one obtains a 5-round GKE $\hat{\pi}$. Our approach for reducing the number of rounds are the two following independent methods:

1. In the first two rounds of $\hat{\pi}$, that is the initialization protocol, each party sends a value *piid* to an initializer. When he received values from each party, he distributes their concatenation within the group. [3] We integrate this procedure into the third round by letting each party sends its *piid* directly to the others and by taking care that these values are inseparable from the remainder of the protocol.
2. In the last round of $\hat{\pi}$, each party broadcasts an acknowledgment. This has two purposes. Firstly, each party can check that all parties obtained the same key. Secondly, it allows straight-line simulatability for the case that one party has already output a group key when another party gets corrupted. We use a non-interactive proof in the penultimate round instead.

As these modifications render the first, second, and last round obsolete, the number of rounds can be reduced by three. Although the approaches might sound quite straightforward, the problems lie (as often) in the security proof.

1. For a proof of security, one has to show that a successful attacker can be used to solve a presumably hard problem. In the case of GKEs, this usually means that a given instance of the problem, for which a solution is sought, is embedded in the

[3] Using randomly generated nonces of the length of the security parameter is one way for choosing unique *piid*s.

messages of the participating parties. But in the case considered here, it is unknown at the beginning which set of parties will eventually participate into a protocol run. Indeed, the number of possible sets of parties (or groups) which might execute a protocol run increases exponentially as the number of parties increases. Thus, new and more sophisticated methods are necessary to embed the problem instance such that the reduction can be shown.
2. While several efficient non-interactive proofs within the random oracle model exist, we aim (to follow the path toward more practicability) for a proof without random oracles.
3. A dedicated simulator which achieves straight-line simulatability is required as the general simulator given by Katz and Shin does not hold anymore.

The idea to solve the first problem deploys a commitment scheme. At the beginning of the protocol execution, each party generates two possible messages m and m' for the first round, together with corresponding internal states s and s'. From these, it chooses *one* internal state, say s, and deletes the other. Then, *both* messages (m and m') are send in the first round together with a commitment to which message is going to be used later (in this example, m). The commitment is assumed to be a perfectly hiding trapdoor bit commitment and its trapdoor is kept secret to everyone. But in the security proof, the ideal adversary $\mathcal{S}$, who chooses the parameters, knows the trapdoor of the commitment scheme and can decommit in such a way that it is appropriate for him.

For the second problem, we make use of pairings on elliptic curves. More precisely, we extend the Burmester-Desmedt protocol over bilinear groups where the validity of messages can be checked by using pairings.

The idea to solve the third problem is to update the internal state in such a way that the key can still be generated but that the initially chosen secret is deleted. This allows to simulate the internal state of the corrupted party instance such that it is in compliance with the revealed group key.

3.2 Protocol Description

In the following, we describe a 2-round GKE π which UC-realizes $\mathcal{F}_{\text{init-gke}}$. Although we prove in Section 4 the security of the protocol is only for the case of an even number of parties, the protocol can easily be extended for an odd number of parties if one or every party plays the role of two parties. The protocol is an extension of the Burmester-Desmedt protocol [12] which additionally uses a bilinear pairing, a perfect hiding trapdoor bit commitment scheme, and a digital signature scheme as further building blocks. We assume that the parties can completely erase their state. This ability is indispensable to ensure forward secrecy, that is corrupting parties does not compromise keys from earlier sessions.

Definition 3. *Let $\mathcal{G}$ and $\tilde{\mathcal{G}}$ denote two cyclic groups of prime order p. A* bilinear pairing *is an efficient mapping $e : \mathcal{G} \times \mathcal{G} \to \tilde{\mathcal{G}}$ such that $e(u^\alpha, v^\beta) = e(u, v)^{\alpha\beta}$ for all $u, v \in \mathcal{G}$ and $\alpha, \beta \in \mathbb{Z}/p\mathbb{Z}$ and there exists $g \in \mathcal{G}$ such that $e(g, g)$ generates $\tilde{\mathcal{G}}$. We will refer to such a tuple $(\mathcal{G}, \tilde{\mathcal{G}}, e, g)$ as a* bilinear pairing quadruple.

Definition 4. *A trapdoor bit commitment scheme is a tuple of algorithms* TGen*,* TCom*,* TVer*, and* TOpen*. Given a value* 1^k*,* TGen *outputs a public parameter* tparam *and the corresponding trapdoor* tdr*. Given* tparam*,* $b \in \{0,1\}$*, and a random tape,* Tcom *outputs* com *and* dec*. Given* tparam, com*, and* dec*,* TVer *outputs* $b \in \{0,1\}$ *or* $\perp$*. Given* tparam, com, dec, tdr, $b' \in \{0,1\}$*,* TOpen *outputs* dec′ *such that* $b' =$ TVer(tparam, com, dec′).

Roughly, a trapdoor bit commitment scheme is hiding/perfect-hiding if distributions of commitments of 0 and 1 are indistinguishable/identical. A trapdoor bit commitment scheme is binding, if given a randomly generated parameter tparam, to output com, dec, dec′ such that $0 =$ TVer(tparam, com, dec) and $1 =$ TVer(tparam, com, dec′) is hard. A trapdoor bit commitment scheme is equivocal if for any (com, dec) $=$ Tcom(tparam, b, r) with $b \in \{0,1\}$, the distributions of (com, dec′ $=$ TOpen(tparam, com, dec, tdr, b')), $b' \in \{0,1\}$, and (com, dec) $=$ Tcom(tparam, b', r') are indistinguishable. Here, r and r' are random tapes. See [25,30] for more details and possible schemes. The trapdoor bit commitment we use here is perfectly hiding, binding, and equivocal.

Definition 5. ***(GKE protocol*** π***)*** *Let* n *denote the size of PID. If we need to distinguish between the sizes of different sets PID, we write* n_{PID} *instead. We index each party in PID by* $i \in \{1, \ldots, n\}$ *in some order and let* Π_i *denote the* i*-th party. For* $\ell \geq n$*, we define* $\Pi_\ell := \Pi_{(\ell \mod n)}$*. Let* $\overline{b}$ *denote* $1 - b$ *for* $b \in \{0,1\}$*. Each party* Π_i *has its public/private key pair* (PK_i, SK_i) *which can be used to sign messages. We assume the deployed digital signature scheme to be existentially unforgeable against chosen message attacks. The public keys are known to all parties. The system parameters are a bilinear pairing quadruple, a randomly chosen element* $v \in \mathcal{G}$*, and parameters* tparam *for a perfect hiding trapdoor commitment scheme. The trapdoor* tdr *that corresponds to* tparam *is kept secret to everyone. The protocol is divided into two communication rounds where each party can send one message to each party per round, and a key generation step at the end. The protocol works as follows:*

Round 1: *When* Π_i *receives a message* $(PID, (\Pi, piid), \text{new-session})$*, it generates a new party instance* $(\Pi_i, piid_i)$*, randomly chooses* $b_i \in \{0,1\}$*, and generates*

$$\begin{aligned} y_{i,b_i} &:= g^{r_i} \text{ with } r_i \in_R \mathbb{Z}/p\mathbb{Z} \\ y_{\overline{b_i}} &\in_R \mathcal{G}, \text{ and} \\ (\text{com}_i, \text{dec}_i) &:= \text{TCom}(\text{tparam}, b_i). \end{aligned}$$

We abbreviate $Y_i := (y_{i,0}, y_{i,1})$*. Remark that* $y_{i,0}$ *and* $y_{i,1}$ *play the role of the two messages we mentioned in the solution approach for problem 1 in the previous sub-section.* $(\Pi_i, piid_i)$ *creates the internal state*

$$st_i^1 := (PID, (\Pi_i, piid_i), Y_i, \text{com}_i, \text{dec}_i, b_i, r_i) \tag{1}$$

and broadcasts

$$(\Pi; PID, (\Pi, piid), Y_i, \text{com}_i). \tag{2}$$

Round 2: *Each instance* (Π_i, piid_i) *receives messages* $(\Pi_j; \mathit{PID}, (\Pi_j, \mathsf{piid}_j), Y_j, \mathsf{com}_j)$ *for* $j \neq i$. *Then it generates*

$$\begin{aligned} \mathit{sid} &:= \{(\Pi_i, \mathsf{piid}_i)\}_{i=1,\ldots,n}, \\ \mathit{cont} &= (\mathit{PID}, \mathit{sid}, Y_1, \mathsf{com}_1, \ldots, Y_n, \mathsf{com}_n), \end{aligned}$$

creates a signature sig_i *on* cont, *and computes for* $\alpha, \beta, \gamma \in \{0, 1\}$

$$x_{i,\alpha,\beta} = \left(\frac{y_{i-1,\alpha}}{y_{i+1,\beta}}\right)^{r_i}, z_{i,\gamma} = e(y_{i+1,\gamma}, v)^{r_i}. \tag{3}$$

We define $X_i := (x_{i,0,0}, x_{i,0,1}, x_{i,1,0}, x_{i,1,1})$ *and* $Z_i := (z_{i,0}, z_{i,1})$. *The internal state is updated to*

$$st_i^2 := ((\Pi_i, \mathsf{piid}_i), cont, b_i, X_i, Z_i)$$

and broadcasts

$$(\Pi_i; (\Pi_i, \mathsf{piid}_i), \mathit{PID}, \mathit{sid}, \mathsf{sig}_i, b_i, \mathsf{dec}_i, X_i)$$

to all parties. Remark that the initial secret r_i *has been deleted and been "replaced" by* X_i *and* Z_i. *While this still allows to compute the key, this step makes the straight-line simulatability feasible (see next section).*

Key generation: *Given an internal state* st_i^2 *and messages*

$$(\Pi_j; \Pi_j, \mathsf{piid}_j, \mathit{PID}, \mathit{sid}, \mathsf{sig}_j, b_j, \mathsf{dec}_j, x_j)$$

for $j \neq i$ *from the second round,* (Π_i, piid_i) *generates the group key as follows. First, it verifies that the following conditions hold for all* $j = 1, \ldots, n$:

1. sig_j *is valid signature of* Π_j *on* cont,
2. $b_j = \mathsf{TVer}(\mathsf{com}_j, \mathsf{dec}_j)$,
3. *the values* $y_{j,b_j}, x_{j,b_{j-1},b_{j+1}}$ *are elements in* $\mathcal{G}$,
4. $e(x_{j,b_{j-1},b_{j+1}}, g) = e(y_{j-1,b_{j-1}}/y_{j+1,b_{j+1}}, y_{j,b_j})$. *Remark that this equation holds only when the parties generated their second messages honestly. This solves the problem 2 we mentioned in the previous sub-section.*

In the positive case (and only then), it generates a group key by

$$\kappa := (z_{i,b_{i+1}})^n e(\prod_{j=1}^{n} (x_{i+j,b_{i+j-1},b_{i+j+1}})^{n+1-j}, v) = e(\prod_{j=1}^{n} g^{r_{j,b_j} r_{j+1,b_{j+1}}}, v), \tag{4}$$

outputs $(\Pi_i, \mathsf{piid}_i, \mathit{PID}, \mathit{sid}, \kappa)$ *and deletes the party instance, i.e., clears the internal state of the party instance* (Π_i, piid_i).

In the above protocol, messages whose length depend on the number n of participants, e.g, *PID* and *sid*, can be shortened to constant size by taking their hash values or commonly agreed alias when possible.

In the next section, we will prove that our protocol UC-realizes $\mathcal{F}_{\text{init-gke}}$. In Appendix A, we will show that any protocol that UC-realizes $\mathcal{F}_{\text{init-gke}}$ needs at least two communication rounds. Hence, our protocol has the minimum possible number of rounds. Furthermore, this shows that the derived bound is tight.

4 Proof of Security

In this section, we prove that the 2-round GKE π proposed in Section 3 UC-realizes $\mathcal{F}_{\text{init-gke}}$. The proof of security relies on a new hardness assumption which we call the linear oracle bilinear Diffie-Hellman (LO-BDH) assumption. Observe that the main reason for introducing the new assumption is not to capture any new security properties but to show that it is indeed possible to construct protocols which achieve the minimum number of rounds and the according minimal number of messages. This assumption is a combination of the decision bilinear Diffie-Hellman assumption and the linear Diffie-Hellman assumption [7].

Definition 6. *We say that the* linear oracle bilinear decision Diffie-Hellman (LO-BDH) assumption holds for a bilinear pairing quadruple $(\mathcal{G}, \tilde{\mathcal{G}}, e, g)$*, if for every polynomial-time adversary* $\mathcal{A}$ *and for every polynomial-size* q*, the advantage of* $\mathcal{A}$ *to win the following game is negligible.*

Setup: *Challenger* $\mathcal{C}$ *randomly chooses* $\alpha, \beta, \gamma, \delta \in \mathbb{Z}/p\mathbb{Z}$*, and* $b \in \{0,1\}$*. Then,* $\mathcal{C}$ *generates* $\chi = e(g,g)^{\alpha\beta\gamma}$ *if* $b = 0$*, otherwise* $\chi = e(g,g)^{\delta}$*.* $\mathcal{C}$ *also randomly chooses* $\mu_i \in \mathbb{Z}/p\mathbb{Z}$ *and generates* $m_i = g^{\mu_i}$ *for* $i = 1, \ldots, q$*.*

Input *At the beginning of the game,* $\mathcal{C}$ *gives* $\mathcal{A}$ *the tuple* $(g, g_1, g_2, g_3, \chi) = (g, g^{\alpha}, g^{\beta}, g^{\gamma}, \chi)$ *and* $\{m_i\}_{i=1,\ldots,q}$

LDH Oracle query *The game consists of* q *rounds. In each round,* $\mathcal{A}$ *can send a query* (i, b_i) *for some* $i \in \{1, \ldots, q\}$ *(but for each* i *only once) and* $b_i \in \{0,1\}$*.*

- *If* $b_i = 0$*,* $\mathcal{C}$ *gives* $g^{\alpha(\mu_i+\beta)}$ *to* $\mathcal{A}$*.*
- *If* $b_i = 1$*,* $\mathcal{C}$ *gives* μ_i *to* $\mathcal{A}$*.*

Answer: *At the end of the game,* $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*. The advantage of* $\mathcal{A}$ *is* $|Pr[b = b'] - 1/2|$*.*

In short, an adversary is required to solve the bilinear Diffie-Hellman problem while it is helped by asking the LDH oracle either to solve a linear Diffie-Hellman problem with respect to given random elements m_i, g_1 and g_2 or to reveal $\log_g m_i$. Although the assumption that an adversary has an access to oracles might appear to be quite strong, several assumptions have been proposed before that allow such queries, e.g., one-more-discrete-log assumption [3], one-more-RSA-inversion assumption [2], LRSW assumption in [14], etc. Furthermore, one can show that LO-BDH assumption assumption holds in the generic bilinear group model (see the full version for a proof).

In [17], the notion of generalized UC (GUC) is proposed which overcomes a weakness of the conventional UC framework. The major motivation for introducing GUC was that UC does not provide deniability and adaptive soundness when a global setup is assumed. A deniable protocol allows a party A to interact with party B in a way that prevents B from later convincing party C that the interaction took place. Adaptive sound arguments maintain the soundness even when party a A proves to a party B that x is in some language L that is related to the global setup. Our scheme assumes the existence of a PKI and a trapdoor commitment scheme for which no one knows its trapdoor. These are global setups that GUC concerns. However, neither deniability nor adaptive soundness play a role in the security requirements for GKEs. We consider GKE to be secure even if a party A can prove to others that A has exchanged a key. We do not

need to concern adaptive soundness since nothing related to globally setup parameter is proved in our protocol. Therefore, we analyze our protocol within the conventional UC framework and consider the GUC framework to be out of scope.

To prove the security of π within the UC framework, we have to describe for any pair of environment $\mathcal{Z}$ and adversary $\mathcal{A}$ an ideal adversary $\mathcal{S}$ such that the real and ideal executions are indistinguishable. Next, we give first a description of $\mathcal{S}$. After that, we sketch the proof which shows that the existence of a successful pair of environment and adversary would contradict the LO-BDH assumption. Due to space limitations, the full proof is can be found in the full version.

Definition 7. *The ideal adversary $\mathcal{S}$ is assumed to have black box access to the adversary $\mathcal{A}$. Messages from $\mathcal{Z}$ to $\mathcal{S}$ ($\mathcal{Z}$ believes it is sending to $\mathcal{A}$) are forwarded to $\mathcal{A}$, and messages from $\mathcal{A}$ to $\mathcal{S}$ ($\mathcal{A}$ believes it is sending to $\mathcal{Z}$) are forwarded to $\mathcal{Z}$. Additionally, $\mathcal{S}$ simulates the real parties. At the beginning, it generates public/private key pairs (PK_Φ, SK_Φ) for each party Φ and gives the public keys to $\mathcal{A}$. Furthermore, it runs* TGen *to create pairs* (tparam, tdr) *and hands the public parameter* tparam *to $\mathcal{A}$. When $\mathcal{S}$ receives a message* $(PID, (\Phi, piid),$ new-session$)$ *from $\mathcal{A}$ for an uncorrupted party Φ, it begins simulating for $\mathcal{A}$ a party instance with party instance ID piid being run for Φ. More precisely, it executes on behalf of $(\Phi, piid)$ the computations for the first round. Any messages sent by $\mathcal{A}$ to Φ are processed by a party instance $(\Phi, piid)$ for suitable piid and any messages output by a party instance $(\Phi, piid)$ are given to $\mathcal{A}$.*

In addition to the above, the ideal adversary $\mathcal{S}$ proceeds as follows:

SKG: *(Session Key Generation) Assume that a simulated party instance $(\Phi, piid)$ outputs a session key $((\Phi, piid), PID, sid, \kappa)$. If $\mathcal{S}$ has sent before (PID, sid', ok) to $\mathcal{F}_{\text{init-gke}}$ for $sid' \neq sid$ and if there exists an uncorrupted party instance $(\Phi', piid') \in sid \cap sid'$, then $\mathcal{S}$ aborts. If not, $\mathcal{S}$ checks whether any of the party instances $(\Phi', piid') \in sid$ have been corrupted.*

- **SKG.¬cor.:** *If no party instance $(\Phi', piid') \in sid$ has been corrupted, then:*
  - **SKG.¬cor.¬ok:** *If $\mathcal{S}$ has not yet sent (PID, sid, ok) to $\mathcal{F}_{\text{init-gke}}$, then $\mathcal{S}$ checks that it has received all pairs $(\Phi, piid) \in sid$ from $\mathcal{F}_{\text{init-gke}}$. If not, $\mathcal{S}$ aborts. Otherwise, it sends (PID, sid, ok) to $\mathcal{F}_{\text{init-gke}}$, followed by $(PID, sid,$ deliver$, (\Phi, piid))$.*
  - **SKG.¬cor.ok:** *If $\mathcal{S}$ has already sent the message (PID, sid, ok) to $\mathcal{F}_{\text{init-gke}}$, this means that another party instance $(\Phi', piid')$ has generated a session key κ' before. If $\kappa \neq \kappa'$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ sends $(PID, sid,$ deliver$, (\Phi, piid))$ to $\mathcal{F}_{\text{init-gke}}$.*
- **SKG.cor.:** *If some party instances $C \subseteq sid \setminus \{(\Phi, piid)\}$ are corrupted, then:*
  - **SKG.cor.¬ok:** *If $\mathcal{S}$ has not yet sent (PID, sid, ok) to $\mathcal{F}_{\text{init-gke}}$, it sends the message $(PID, (\Phi', piid'),$ new-session$)$ to $\mathcal{F}_{\text{init-gke}}$ on behalf of all corrupted party instances $(\Phi', piid') \in C$ who have not done so already. If $\mathcal{S}$ does not receive $(PID, (\Phi', piid'))$ for* all *pairs $(\Phi', piid') \in sid$ after executing the above, it aborts. Otherwise, it sends the messages (PID, sid, ok), $(PID, sid,$ key$, \kappa)$, and $(PID, sid,$ deliver$, (\Phi, piid))$ to $\mathcal{F}_{\text{init-gke}}$.*
  - **SKG.cor.ok:** *If $\mathcal{S}$ has already sent (PID, sid, ok) to $\mathcal{F}_{\text{init-gke}}$, then $\mathcal{S}$ has sent immediately after $(PID, sid, \kappa',$ set$)$ to $\mathcal{F}_{\text{init-gke}}$. If $\kappa' \neq \kappa$ then $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ sends $(PID, sid,$ deliver$, (\Phi, piid))$ to $\mathcal{F}_{\text{init-gke}}$.*

COR: *When $\mathcal{A}$ intends to corrupt a party instance (Φ, piid), $\mathcal{S}$ provides $\mathcal{A}$ with the current internal state as follows:*

COR.¬ok: *If $\mathcal{S}$ has not yet sent to $\mathcal{F}_{\text{init-gke}}$ the message (PID, sid, ok) for some (PID, sid) with $(\Phi, \mathsf{piid}) \in sid$, then $\mathcal{S}$ simply gives $\mathcal{A}$ the current internal state of (Φ, piid). Such a state is guaranteed to exist unless some message (PID, sid, ok) has been sent followed by $(PID, sid, \mathsf{deliver}, \Phi, \mathsf{piid})$. This state can be either one of the two internal states $st^i_{\Phi,\mathsf{piid}}$ for $i = 1, 2$, or the empty state, depending on the current round of the protocol execution sid.*

COR.ok: *Consider now the case that $\mathcal{S}$ has already sent (PID, sid, ok) to $\mathcal{F}_{\text{init-gke}}$.*

COR.ok.¬del: *If $\mathcal{S}$ has not yet sent $(PID, sid, \mathsf{deliver}, (\Phi, \mathsf{piid}))$ (with $(\Phi, \mathsf{piid}) \in sid$), then it checks if (Φ, piid) does have a internal state $st^2_{(\Phi,\mathsf{piid})}$, i.e., the second round is completed. If this is* not *the case, then $\mathcal{S}$ aborts.*

Otherwise, let the party instances be ordered as explained in the protocol description in Section 3, that is $sid = \{(\Phi_1, \mathsf{piid}_1), \ldots, (\Phi_n, \mathsf{piid}_n)\}$, and let $(\Phi_i, \mathsf{piid}_i)$ denote the party instance that is corrupted by $\mathcal{S}$ to obtain a key (PID, sid, κ) from $\mathcal{F}_{\text{init-gke}}$. This might either be provided by $\mathcal{S}$ before or be generated by $\mathcal{F}_{\text{init-gke}}$. In the first case, $\mathcal{S}$ simply forwards the internal state of $(\Phi_i, \mathsf{piid}_i)$ to $\mathcal{A}$. For the second case, we make use of the fact that as $\mathcal{S}$ has already sent (PID, sid, ok). This implies that at least one simulated copy $(\Phi_j, \mathsf{piid}_j) \in sid$ has already distributed values (b_j, X_j) and *has received $n-1$ tuples (b_ℓ, X_ℓ) before. $\mathcal{S}$ uses these values and κ, which it obtains by corrupting a party instance via $\mathcal{F}_{\text{init-gke}}$, to replace the value of $z_{i,b_{i+1}}$ in the internal state of $(\Phi_i, \mathsf{piid}_i)$ to*

$$z'_{i,b_{i+1}} := \left(\kappa / e(\prod_{j=1}^{n} ((x_{i+j,b_{i+j-1},b_{i+j+1}})^{n+1-j}, v) \right)^{1/n},$$

sets $z'_{i,1-b_{i+1}} := z_{i,1-b_{i+1}}$, and hands to $\mathcal{A}$ the internal state

$$(PID, sid, (\Phi_i, \mathsf{piid}_i), cont_i, \mathsf{dec}_i, b_i, X_i, Z'_i).$$

COR.ok.del: *If $\mathcal{S}$ has already sent $(PID, sid, \mathsf{deliver}, \Phi, \mathsf{piid})$ to $\mathcal{F}_{\text{init-gke}}$, then $\mathcal{S}$ returns nothing (i.e., an empty internal state) to $\mathcal{A}$.*

Theorem 1. *The GKE π UC-realizes $\mathcal{F}_{\text{init-gke}}$ under the LO-BDH assumption when the number n of participants is even.*

Proof

(Sketch) We sketch only the proof ideas here. A detailed proof can be found in the full version. The description of $\mathcal{S}$ contains several cases in which $\mathcal{S}$ aborts. Using a hybrid argument, one can prove that these abortions never occur unless the used signature scheme or commitment scheme are broken.

With respect to the straight-line simulatability, an ideal adversary can easily simulate the internal state of the (simulated) party instances in the case of corruption if no

party instance has output a key. However, if the adversary $\mathcal{A}$ requests the corruption of a party instance when some other party instances have output keys before, the ideal adversary $\mathcal{S}$ is forced to present an internal state to $\mathcal{A}$ for the corrupted party instance that fits to the key that has been randomly chosen by the ideal functionality $\mathcal{F}_{\text{init-gke}}$. The simulator description tells how this can be achieved and in the full proof, we show that this reconstructed internal state cannot be distinguished from a honestly generated state if the LO-BDH assumption holds.

Finally, we prove that if real and ideal executions are distinguishable, then the LO-BDH assumption is violated. For this purpose, we embed a corresponding problem into the executions. For this embedding, we make use of the fact that although the number of possible sets *sid* is superpolynomial in the number of invoked party instances, the number of sets *sid* that actually appear during the execution is only polynomial in the security parameter. For this purpose, we make use of the trapdoor in the commitment scheme.

5 Conclusions and Open Questions

In this paper, we followed the path initiated by Katz and Shin in [35] and presented a group key exchange protocol that is secure within the universal composability framework. However, while Katz and Shin demonstrated merely the general feasibility of constructing UC-secure GKEs, we aimed for improved solutions. Our protocol requires only two communication rounds (which we show to be minimal) and includes session identifier generation (which is a prerequisite within the UC-framework but is only rarely provided in practice). In the protocol, every party send to every other party in each round one message, which yields a total number of $2n(n-1)$ messages. We show that the number of messages cannot be further reduced in two-round protocols.

Although our protocol improves over existing results, there is still room for further improvements or extensions. An important question is if a two-round UC-realization of $\mathcal{F}_{\text{init-gke}}$ exists where the security can be reduced to a more standard assumption. Furthermore, we imagine that the basic idea on how to integrate protocol initialization into the protocol (using a trapdoor commitment scheme) might be transferred to other types of protocols as well.

References

1. Barak, B., Lindell, Y., Rabin, T.: Protocol initialization for the framework of universal composability. Cryptology ePrint Archive, Report2004/006 (2004), http://eprint.iacr.org/
2. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. J. Cryptology 16(3), 185–215 (2003)
3. Bellare, M., Palacio, A.: Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)

5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
6. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: STOC, pp. 57–66. ACM, New York (1995)
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin [13], pp. 41–55
8. Bresson, E., Chevassut, O., Pointcheval, D.: Provably authenticated group diffie-hellman key exchange - the dynamic case. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 290–309. Springer, Heidelberg (2001)
9. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic group diffie-hellman key exchange under standard assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
10. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group diffie-hellman key exchange. In: CCS 2001: Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 255–264. ACM Press, New York (2001)
11. Bresson, E., Manulis, M., Schwenk, J.: On Security Models and Compilers for Group Key Exchange Protocols. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 292–307. Springer, Heidelberg (2007)
12. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system (extended abstract). In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
13. Cachin, C., Strobl, R.: Asynchronous Group Key Exchange with Failures. In: Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC 2004), pp. 357–366. ACM Press, New York (2004)
14. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin [31], pp. 56–72
15. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (revised in 2005) (2000), http://eprint.iacr.org/
16. Canetti, R.: Universally composable signature, certification, and authentication. In: CSFW, p. 219. IEEE Computer Society, Los Alamitos (2004)
17. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (2007)
18. Canetti, R., Fischlin, M.: Universally composable commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
19. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005)
20. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
21. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002)
22. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. J. Cryptology 19(2), 135–167 (2006)
23. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)
24. Canetti, R., Rabin, T.: Universal composition with joint state. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 265–281. Springer, Heidelberg (2003)

25. Crescenzo, G.D., Katz, J., Ostrovsky, R., Smith, A.: Efficient and non-interactive non-malleable commitment. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 40–59. Springer, Heidelberg (2001)
26. Desmedt, Y.G., Pieprzyk, J., Steinfeld, R., Wang, H.: A Non-Malleable Group Key Exchange Protocol Robust Against Active Insiders. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 459–475. Springer, Heidelberg (2006)
27. Dutta, R., Barua, R., Sarkar, P.: Provably secure authenticated tree based group key agreement. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 2004. LNCS, vol. 3269, pp. 92–104. Springer, Heidelberg (2004)
28. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
29. Fischlin, M.: Universally composable oblivious transfer in the multi-party setting. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 332–349. Springer, Heidelberg (2006)
30. Fischlin, M., Fischlin, R.: Efficient non-malleable commitment schemes. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 413–431. Springer, Heidelberg (2000)
31. Franklin, M. (ed.): CRYPTO 2004. LNCS, vol. 3152. Springer, Heidelberg (2004)
32. Hofheinz, D., Müller-Quade, J., Steinwandt, R.: Initiator-resilient universally composable key exchange. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 61–84. Springer, Heidelberg (2003)
33. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions. In: STOC 1989: Proceedings of the twenty-first annual ACM symposium on Theory of computing, pp. 12–24. ACM Press, New York (1989)
34. Impagliazzo, R., Zuckerman, D.: How to recycle random bits. In: FOCS, pp. 248–253. IEEE, Los Alamitos (1989)
35. Katz, J., Shin, J.S.: Modeling insider attacks on group key-exchange protocols. In: CCS 2005: Proceedings of the 12th ACM conference on Computer and communications security, pp. 180–189. ACM Press, New York (2005)
36. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. J. Cryptol. 20(1), 85–113 (2007)
37. Kurosawa, K., Furukawa, J.: Universally composable undeniable signature. Cryptology ePrint Archive, Report 2008/094 (2008), `http://eprint.iacr.org/`
38. Le, T.V., Burmester, M., de Medeiros, B.: Universally composable and forward-secure rfid authentication and authenticated key exchange. In: Bao, F., Miller, S. (eds.) ASIACCS, pp. 242–252. ACM, New York (2007)
39. Wikström, D.: A universally composable mix-net. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 317–335. Springer, Heidelberg (2004)

A Lower Bounds on the Communication

In this section, we prove that any GKE that UC-realizes $\mathcal{F}_{\text{init-gke}}$ (short: a INIT-GKE) needs at least two communication rounds and that any two-round GKE that UC-realizes $\mathcal{F}_{\text{init-gke}}$ requires at least $2n(n-1)$ messages.[4] As the protocol proposed in Section 3

[4] We count messages which are send to different parties separately. That is a broadcast message would count as $n-1$ messages as it is received by $n-1$ parties.

meets these lower bounds, the bound derived on the number of rounds are tight and the protocol has minimum communication effort.

As said before, we assume that the parties are *inter-session memoryless*, that is no information from one session can be used for further sessions. A party who has an inter-session memory can memorize all SID of protocols in which it was involved and can deny to take part in a protocol of the same SID even when those SIDs are given by others. The reason for this assumption is that we would like to model the parties as relaxed as possible. That is we would not like to demand from parties for example to keep track of all used *sid*s and so on. Of course, stronger parties which might have inter-session memory might run more efficient protocols.

Theorem 2. *If a 2-round protocol UC-realizes the INIT-GKE functionality $\mathcal{F}_{\text{init-gke}}$ and if all participating parties are inter-session memoryless, then in each round every party sends a messages to every other party of the same group.*

The theorem follows from the following two lemmas 1 and 2. Both lemmas use the fact that in the case of the ideal executions, it can never happen that a party instance of a honest party will take part in two different protocol executions (or sessions) which eventually generate a key. The reason is that we assumed that party instances of honest parties always choose a unique *piid*. Once a protocol execution starts (more precisely: $\mathcal{F}_{\text{init-gke}}$ starts the key generation), the identity of the party instance is deleted from the list of ready instances. As we suppose authenticated channels, an attacker can not re-request a protocol execution on behalf of the same party instance as long as the party is uncorrupted. Hence, there is now way to put this party instance back on the list of ready instances.

Lemma 1. *An INIT-GKE π does not UC-realize $\mathcal{F}_{\text{init-gke}}$ if there exist parties Φ and Φ' such that with non negligible probability Φ does not send a message to Φ' in the* second *round.*

Proof. Let π be a INIT-GKE for which the mentioned condition holds. The following adversary $\mathcal{A}$ is able to distinguish the real and the ideal executions which shows that π cannot UC-realize $\mathcal{F}_{\text{init-gke}}$.

1. Consider a protocol execution where *sid* denotes the set of participating party instances with a least two uncorrupted instances. Adversary $\mathcal{A}$ randomly selects two uncorrupted party instances $(\Phi, piid), (\Phi', piid')$ and corrupts the remaining party instances $cpid := sid \setminus \{(\Phi, piid), (\Phi', piid')\}$.
2. $\mathcal{A}$ honestly takes part in the protocol execution on behalf of the corrupted parties in *cpid*. Let $m(1, (\Phi, piid), (\Phi', piid'))$ denote the message $(\Phi, piid)$ sends to $(\Phi', piid')$ and $m(1, (\Phi, piid), cpid)$ denote the collection of all messages $(\Phi, piid)$ sends to *cpid* in the first round. These messages might be empty if no messages have been send. The adversary completes the protocol and checks if $(\Phi, piid)$ does have to send a message to $(\Phi', piid')$ in the second round. If this is the case, it repeats this procedure again. In the other case, what happens with non-negligible probability by assumption, it proceeds to the next step and $(\Phi', piid')$ outputs a key as usual.

3. Once the adversary obtained $m(1, (\Phi, \mathit{piid}), (\Phi', \mathit{piid}'))$ and $m(1, (\Phi, \mathit{piid}), \mathit{cpid})$, we show in the following that he can start a second protocol execution in which (Φ, piid) participates again. As described above, this cannot happen in ideal executions.
   (a) The adversary $\mathcal{A}$ starts a new protocol with the same group of parties. While $\mathcal{A}$ lets Φ' create a new party instance (Φ', piid''), it re-uses for the party instances in *cpid and* (Φ, piid) the messages from the previous protocol executions. Observe that (Φ, piid) takes part at two protocol executions while Φ is assumed to be honest. We will show now that this second execution will eventually lead to a key generation as well. Since (Φ') is inter-session memoryless, it cannot detect that the party instances *cpid* and (Φ, piid) have used exactly the same messages before. Hence, (Φ', piid'') completes the first round without any problem.
   (b) From the messages (Φ', piid'') has sent to *cpid* in the first round, $m(1, (\Phi, \mathit{piid}), \mathit{cpid})$, and the random tapes inputs to *cpid*, the adversary can generate all messages the parties in *cpid* have to send to (Φ', piid'') in the second round. By assumption, with non negligible probability (Φ, piid) is not required to send any message to (Φ', piid'') in the second round. In this case, everything is alright from the point of view of (Φ', piid'') which outputs a group key.

Concluding, this leads to a second protocol execution which eventually leads to a group key although the same party instance (Φ, piid) participated in both runs. As explained above, this is not possible in ideal executions. Therefore, the protocol does not UC-realize $\mathcal{F}_{\text{init-gke}}$. □

Lemma 2. *An INIT-GKE protocol does not UC-realizes* $\mathcal{F}_{\text{init-gke}}$ *if there exist participants* (Φ, piid) *and* (Φ', piid') *such that with non negligible probability they may send messages to each other in the* second *round but* (Φ', piid') *does not send a message to* (Φ, piid) *in the* first *round*

Proof. The proof is very similar to the proof of Lemma 2 and can be found in the full version.

Corollary 1. *Every universally composable INIT-GKE protocol requires at least two round.*

Corollary 2. *The most efficient case in two round protocols is when every party broadcasts one message in each round.*

An Identity-Based Key Agreement Protocol for the Network Layer

Christian Schridde, Matthew Smith, and Bernd Freisleben

Department of Mathematics and Computer Science, University of Marburg
Hans-Meerwein-Str. 3, D-35032 Marburg, Germany
{schriddc,matthew,freisleb}@informatik.uni-marburg.de

Abstract. A new identity-based key agreement protocol designed to operate on the network layer is presented. Endpoint addresses, namely IP and MAC addresses, are used as public keys to authenticate the communication devices involved in a key agreement, which allows us to piggyback much of the security overhead for key management to the existing network infrastructure. The proposed approach offers solutions to some of the open problems of identity-based key agreement schemes when applied to the network layer, namely multi-domain key generation, key distribution, multi-domain public parameter distribution, inter-domain key agreement and network address translation traversal.

1 Introduction

Current network security protocols like IPsec use either pre-shared keys or a Public Key Infrastructure (PKI) to secure the communication channel. The pre-shared keys approach is suitable for small networks but does not scale well. The PKI approach scales better but has a high management overhead [27],[2]. To avoid the complexity of authenticated public key distribution, Shamir [25] in 1984 proposed the concept of identity-based cryptography (IBC) which allows an arbitrary string to be used as a public key. Since then, several identity-based encryption (IBE) schemes [16] and identity-based key agreement protocols [14] have been suggested, but it was not until 2001 when the first practical IBE systems were introduced by Boneh and Franklin using Weil pairings [6] and Cocks using quadratic residues [10].

IBC has been applied to several application layer protocols, with the main focus lying on e-mail protection. Some attempts have been made to apply IBC to lower layers like the network layer to offer lightweight alternatives to PKI based security solutions, but a number of problems have hindered the adoption of IBC at this level. Unlike the application layer where identifiers are usually unique (like e-mail or SIP addresses) and do not change owners, IP addresses can both change owners on a regular basis and are not necessarily unique. For both scarcity and security reasons, many devices have private IP addresses and access the Internet using the Network Address Translation (NAT) protocol. This creates problems for IBC, since outside the private network the device behind the NAT router has a different identifier, namely that of the NAT router. Furthermore, the private identifier of the "NATed" device is most likely used by other resources in other private networks and thus is not unique. Another practical issue when deploying

R. Ostrovsky, R. De Prisco, and I. Visconti (Eds.): SCN 2008, LNCS 5229, pp. 409–422, 2008.

an IBC system on the network layer is that many different organizations are responsible for different parts of the Internet and thus it is unlikely that a single trusted authority can be found to operate the identity private key generator (ID-PKG). Several solutions have been proposed which allow multiple ID-PKGs to interoperate [15,17,7,5], but these systems require either cooperation between the ID-PKGs or a hierarchical approach with a trusted party at the top. Both approaches are difficult to use in the Internet due to organizational difficulties and conflicting business interests. As demonstrated by approaches based on a Certificate Authority (CA), there will always be competing organizations offering the same service for the same protocol (e.g. signing RSA public keys) without wanting to cooperate on the corporate level. Thus, to successfully deploy IBC on the network layer, the IBC system must be able to cope with NAT, address reuse (and consequently dynamic identity key deployment), and it must allow competing organizations to operate their ID-PKG independently of other ID-PKGs while still enabling cross-domain execution of the IBC protocols for their customers.

In this paper, a new identity-based key agreement protocol is introduced which focuses on the issues to be solved when implementing IBC on the network layer. The proposed approach is realized using IP addresses on the network layer and optionally MAC addresses on the data link layer for bootstrapping purposes. It utilizes the mathematics also used in the traditional Diffie-Hellman [12] key agreement and Rivest-Shamir-Adleman (RSA) [23] public key cryptography approaches, and in the key distribution system proposed by Okamoto [19]. Solutions to the problems of multi-domain key generation, key distribution, multi-domain public parameter distribution, cross-domain key agreement and NAT are presented.

The paper is organized as follows. Section 2 discusses related work. In Section 3, the new identity-based key agreement scheme is presented. Section 4 discusses implementation issues. Section 5 concludes the paper and outlines areas for future research.

2 Related Work

Since Shamir's pioneering IBC proposal in 1984, various IBE systems have been proposed. In 2001, Boneh and Franklin [6] introduced an operational IBE system based on bilinear pairings on elliptic curves. By using efficient algorithms to compute the pairing function, the system can be used in real world applications. Its main application focus is e-mail, dealing with key distribution and expiration in this domain. Several optimizations and extensions of Boneh and Franklin's IBE approach have been suggested (e.g. [17], [8], [9], [26]), all based on the Weil pairings [6] originally used by Boneh and Franklin. These extensions include hierarchical IBE systems [15] and public parameter distribution systems [27]. However, the main focus of all proposals is on application level security, and e-mail is the main application.

Apart from the full IBE systems, several identity based key agreement schemes have been proposed, such as [14], [19], [17], [24], [9], [26] and [8]. For example, the key distribution system proposed by Okamoto [19] extracts identity information and combines it into a session initiation key in a similar manner as in our scheme, but does not address the problem of key agreement between different domains. Other of these approaches offer solutions to combine a number of ID-PKGs, but not in an independent manner.

Appenzeller and Lynn [2] have introduced a network layer security protocol using IBC. The protocol does not deal with the setup phase, i.e. public parameter distribution, identity key distribution, and is not compatible with NAT routers. In [27], a working solution for the problem of public parameter distribution on an Internet scale using the Domain Name System (DNS) has been presented. However, the paper does not deal with the critical issues of identity key distribution or NAT traversal.

In all of the above proposals, an arbitrary string can be used as an identifier and thus also as a public key. However, since the main application focus of most of these proposals is e-mail, e-mail addresses are the only identifiers for which problems and solutions are discussed in depth. Problems occurring in other areas, such as IP, Voice-over-IP, NAT traversal, multi-provider networks and dynamic IP redistribution, are not discussed. Furthermore, although hierarchical Identity Private Key Generator (ID-PKG) systems have been introduced [15,17,7,5], they usually require either a root authority trusted by all ID-PKGs, or that the different ID-PKGs cooperate during their setup. This is problematic both in the Internet and in telecommunication networks, since no single trusted authority exists, and service providers are typically competitors.

Two other approaches have been suggested in the context of secure IPv6 network protocols that are relevant to this paper. The Host Identity Protocol (HIP) [18] removes the need for binding IP addresses to public keys using certificates by creating a completely new form of addresses, namely HIP addresses which are constructed by hashing the public key of a device. This creates two requirements which a HIP system must meet: (a) the public keys must be created before the address allocation can be performed and (b) a new protocol layer must be implemented between the transport and network layer which maps HIP identifiers to the routable IPv6 addresses and provides authentication. To address the last critical issue, Cryptographically Generated Addresses (CGA) [4] have been propsed to encode a public key into the 64-bit identifier of the IPv6 address, thus avoiding the need to change the protocol stack. However, CGA still requires the public key to be created before the IPv6 address and restricts the choice of addresses which can be used. Obviously, getting ISPs to issue particular IPv6 addresses based on user keys is a difficult task.

3 A New Identity-Based Key Agreement Scheme

3.1 Requirements

The requirements that should be met by the proposed key agreement protocol for the network layer are as follows:

– Complexity reduction: Contemporary networks are already complex environments with thousands of network enabled devices coming and going, roaming in foreign networks and changing providers on a continuous basis. A security infrastructure should not double the administrative effort by creating one or more mirrors of the administrative infrastructure for public key certification or storage. At some point, keys or public parameters must be distributed if authentication is required, but these key management issues should be kept as local as possible while offering global interaction.

- Multi-provider operation: The Internet is driven by multi-organizational entities which will not adopt a single trusted third party or a single trust hierarchy. Therefore, a cryptographic system must allow multiple security providers to inter-operate with minimal requirements.
- NAT traversal: NAT is a common occurrence in the Internet, and even with the adoption of IPv6 it will not disappear completely, since many organizations use NAT to protect networked resources as well as to conserve public IP addresses.
- Dynamic/transient endpoint addresses: Due to the relatively small number of IPv4 addresses, these endpoint addresses are shared or reused after a certain amount of time. A cryptographic system must be able to deal with changing owners of endpoint addresses.

To combine the ease of use of a Diffie-Hellman key agreement protocol with the security of an authenticated RSA key exchange, we present an identity-based key agreement protocol in which the endpoint addresses of the communication devices are used to authenticate the devices involved in a key agreement. The use of IBC allows us to both reduce and spread the administrative burden of key management by seamlessly integrating the cryptographic solution into the existing network infrastructure. Our scheme allows multiple organizations to operate ID-PKGs autonomously, while allowing their customers to operate across organizational borders. This greatly simplifies the otherwise thorny issue of private key distribution present in global IBE or PKI solutions. Furthermore, the choice of coupling the cryptographic system to the network layer allows us to piggyback much of the security overhead to the existing network infrastructure. The proposed system is capable of coping with NAT traversal, and using key expiration coupled with dynamic key deployment allows for dynamic endpoint allocation.

In the following, our identity-based key agreement system called Secure Session Framework (*SSF*) is presented. First, the four basic steps of the system are described for the single ID-PKG scenario. Then, the scheme is extended to incorporate multiple autonomous ID-PKGs which use different public parameters without requiring entities from one ID-PKG's domain to contact the ID-PKGs of other domains.

3.2 Algorithmic Overview

The proposed identity-based key agreement protocol *SSF* consists of four main algorithms: **Setup**, **Extract**, **Build SIK**, and **Compute**. The first two are executed by the private key generator and the last two are executed by a participating device. Their description follows below.

3.3 Key Agreement

The **Setup** algorithm is executed by the ID-PKG. This part of the key agreement protocol is only performed once and creates both the master secrets P and Q as well as the public parameters.

Setup - The **Setup** algorithm is executed by the ID-PKG.
Input: $k \in \mathbb{N}$
Step 1: Choose an arbitrary integer $R > 1$ from $\mathbb{Z}^+$.
Step 2: Generate two primes, P and Q, of bit length k with the following properties:
1. The prime factorization of $(P-1)$ contains a large prime P'
2. The prime factorization of $(Q-1)$ contains a large prime Q'
3. $(\varphi(PQ), R) = 1$, where $\varphi(\cdot)$ is the Totient Function.
4. The computation of discr. log. must be infeasible in $\mathbb{Z}_P$ and $\mathbb{Z}_Q$.
Step 3: Compute the product $N = PQ$
Step 4: Choose a generator G of a subgroup $\mathbb{G}$ of $\mathbb{Z}_N$ whose order contains at least one of the primes P' or Q'.
Step 5: Choose a cryptographic collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_\mathbb{N}$.
Output: $PSP = (N, G, R, H(\cdot))$, $SP = \{P, Q\}$

Definition 1 (Public, Shared Parameters). *The public, shared parameters (PSP) of a domain* ***D*** *of the key agreement scheme* SSF *is the quadruple* $PSP = (N, G, R, H(\cdot))$

The **Extract** algorithm creates the identity key (i.e. the private key) for a given identity. This algorithm is executed by the ID-PKG. If all IDs are known and the range is not too big (e.g. a Class B or C subnet of the Internet), it is possible to execute this step for all IDs offline, and the master secrets can then be destroyed, if required.

Extract - The **Extract** algorithm is executed by the ID-PKG.
Input: PSP, SP, ID
Let ID be a given identity. The algorithm computes $d_{ID} \equiv H(ID)^{1/R}\ mod\ N$. The integer d_{ID} is called the *identity key* and is given to the entity E_{ID}.
Note: Due to the requirement $(\varphi(N), R) = 1$, the R-th residues form a permutation in $\mathbb{Z}_\mathbb{N}$.
Ouput: d_{ID}

The **Build SIK** algorithm is executed by the devices taking part in the key agreement.

Build SIK - The **Build SIK** algorithm is executed by the entity E_{ID}
Input: PSP, d_{ID}, k_1, k_2
Step 1: Choose a random integer r_{ID} in the interval $[2^{k_1}, 2^{k_2}]$.
Step 2: Compute $SIK_{ID} \equiv G^{r_{ID}} \cdot d_{ID}\ mod\ N$.
SIK_{ID} is the SIK (session initiation key) for the identity string ID which belongs to entity E_{ID}.
Output: SIK_{ID}

The random integer r_{ID} is generated with a secure number generator to make r_{ID} unpredictable. The private identity key is used in combination with this randomly chosen integer and the generator in such a way that it is not possible to extract the identity key from the SIK. This is due to the fact that the multiplications are performed in the ring $\mathbb{Z}_\mathbb{N}$ and the result set of a division in the ring $\mathbb{Z}_\mathbb{N}$ is so large that the extraction of the

identity key is infeasible. The SIK is then sent over an unsecured channel to the other party and vice versa. The SIK must be greater than zero to prevent a trivial replacement attack where an attacker replaces the SIKs with zero which in turn would make the session key zero as well. Any other replacement attacks lead to invalid session keys.

The final step of the key agreement process is the computation of the session key using the **Compute** algorithm which is executed by the devices taking part in the key agreement. By applying the inverse of the hash value of the opposite's identity, the involved identity key is canceled out. Only if both endpoint addresses match their identity keys, a valid session key is created.

Compute - The **Compute** algorithm is computed when two entities are performing a key agreement.
Input for E_{ID_A}: PSP, $SIK_{ID_B} > 0, ID_B, r_{ID_A}$
Input for E_{ID_B}: PSP, $SIK_{ID_A} > 0, ID_A, r_{ID_B}$
When E_{ID_A} receives the session initiation key from E_{ID_B}, it calculates
$(SIK_B^R H(ID_B)^{-1})^{r_{ID_A}} \equiv ((G^{r_{ID_B}} d_{ID_B})^R H(ID_B)^{-1})^{r_{ID_A}} \equiv G^{Rr_{ID_A} r_{ID_B}} \equiv S \bmod N$
When E_{ID_B} receives the session initiation key from E_{ID_A},it calculates
$(SIK_A^R H(ID_A)^{-1})^{r_{ID_B}} \equiv ((G^{r_{ID_A}} d_{ID_A})^R H(ID_A)^{-1})^{r_{ID_B}} \equiv G^{Rr_{ID_A} r_{ID_B}} \equiv S \bmod N$
Output: S

3.4 Key Agreement between Different Domains

The ID-PKG determines the public, shared parameters, and all entities which receive their identity key for their IDs from this generator can establish a key agreement among each other. In practice, it is unlikely that all devices will receive their identity key from the same security domain, since this would imply the existence of a third party trusted by all with a secure communication link to all devices. The Internet is divided into many Autonomous Systems (AS) each with its own IP range and responsible for the management of its users. Thus, it is desirable that each AS can operate its own ID-PKG.

In this section, we show how cross-domain key agreement can be achieved such that only the public parameters must be distributed (which will be discussed in section 4). Each device only needs a single identity key, and the ID-PKGs do not need to agree on common parameters or participate in any form of hierarchy. In the following, we assume without loss of generality, that there are two domains D_1 and D_2. Their public parameters are $(N_1, G_1, R_1, H_1(\cdot))$ and $(N_2, G_2, R_2, H_2(\cdot))$, respectively. Every parameter can be chosen independently. The case that $(R_2, \varphi(N_1)) > 1$ or $(R_1, \varphi(N_2)) > 1$ is not critical, since no R-th roots must be computed regarding the other domain's modulus. The two moduli N_1 and N_2 were chosen according to the requirements stated in the **Setup** algorithm, i.e. the computation of discrete logarithms is infeasible in $\mathbb{Z}_{N_1}$ and $\mathbb{Z}_{N_2}$, respectively. Consequently, an algorithm such as the Pohlig-Hellman algorithm [20] cannot be applied and Pollard's $P-1$ factoring algorithm [21] will not be a threat. Thus, a random non-trivial integer has a large order in $\mathbb{Z}_{N_1 N_2}$ with an overwhelming probability, and the computation of discrete logarithms is infeasible in $\mathbb{Z}_{N_1 N_2}$. In the following, an entity E_{ID_A} from D_1 wants to communicate with E_{ID_B} from D_2.

Cross-Domain Extension (from the view of entity E_{ID})
Input: PSP_1, PSP_2, d_{ID}
Step 1: Calculate the common, shared, public parameters: $PSP_{1,2} = (N_1 \cdot N_2, G_1 \cdot G_2, R_1 \cdot R_2, H_2(\cdot))$.
Step 2: Use the Chinese-Remainder Theorem to calculate the integer $\widetilde{d}_{ID}$: $\widetilde{d}_{ID} \equiv d_{ID} \; mod \; N_1$ and $\widetilde{d}_{ID} \equiv 1 \; mod \; N_2$
Step 3: Use the Chinese-Remainder Theorem to calculate the integer $\widetilde{H_1}(ID)$: $\widetilde{H_1}(ID) \equiv H_1(ID)^{R_2} \; mod \; N_1$ and $\widetilde{H_1}(ID) \equiv 1 \; mod \; N_2$
Output: $SIK_{ID}^{(1,2)}$

In step 1 of the cross-domain extension algorithm, the common shared public parameters are the element-wise product of both sets of domain parameters. In step 2, entity E_{ID_A} extends its identity key using the Chinese-Remainder Theorem. In step 3, entity E_{ID_A} extends its hash identifier also using the Chinese-Remainder Theorem.

The procedure for entity E_{ID_B} is analog, only the indices change from A to B. Key agreement is then performed using the following extension of the original algorithm.

Cross-Domain: Compute algorithm
Input for E_{ID_A}: $PSP_{(1,2)}$, $SIK_{ID_B}^{(1,2)} > 0$, ID_B, r_{ID_A}
Input for E_{ID_B}: $PSP_{(1,2)}$, $SIK_{ID_A}^{(1,2)} > 0$, ID_A, r_{ID_B}
When E_{ID_A} receives the session initiation key from E_{ID_B}, it calculates
$\left(((G_1G_2)^{r_{ID_B}} \widetilde{d}_{ID_B})^{R_1R_2} \widetilde{H_2}(ID_B)^{-1}\right)^{r_{ID_A}} \equiv (G_1G_2)^{R_1R_2 r_{ID_A} r_{ID_B}} \equiv S \; mod \; (N_1N_2)$
When E_{ID_B} receives the session initiation key from E_{ID_A},it calculates
$\left(((G_1G_2)^{r_{ID_A}} \widetilde{d}_{ID_A})^{R_1R_2} \widetilde{H_1}(ID_A)^{-1}\right)^{r_{ID_B}} \equiv (G_1G_2)^{R_1R_2 r_{ID_A} r_{ID_B}} \equiv S \; mod \; (N_1N_2)$
Output: S

A security analysis, correctness proofs and further details on the algorithms can be found in [3].

4 Implementation Issues

In the following, several issues for deploying the proposed system in practice are discussed. It will be shown how the public parameters and the identity keys are distributed in multi-provider scenarios and how dynamic IP addresses are handled. Furthermore, a detailed description of how our system deals with the NAT problem will be given. One of the important issues of any multi-organizational cryptographic system is the distribution of the public parameters and keys. It should be noted that a main requirement is to try to minimize the number of global distribution steps in favor of local distribution steps, since this distributes the workload and reduces the risk of a global compromise. In a scenario with N providers, each with M customers where $M >> N$, we have $N \cdot M$ customers in total. This means that $N \cdot M$ private/identity keys need to be distributed. In a PKI, in the worst case in which everybody wants to communicate with everybody else, $(N \cdot M - 1) \cdot (N \cdot M)$ public keys need to be exchanged and managed. In our system,

only the public parameters of the N providers need to be exchanged. This reduces the number of transfers from $N \cdot M$ local and $(N \cdot M - 1) \cdot (N \cdot M)$ global transfers to $N \cdot M$ local transfers and only N global transfers, and since $M >> N$, this is a large saving. Even using traditional key distribution mechanisms, our system offers a significant saving compared to a PKI in key escrow mode. In the following, further optimizations of the distribution process which are possible due to the network centric approach of our solution will be suggested.

4.1 Distribution of Shared, Public Parameters

Like most other IBC approaches, our system also uses shared public parameters. In a single domain scenario, the distribution of the public parameters is straightforward. However, if each AS runs its own ID-PKG, the number of public parameters and the binding between public parameters and identity keys becomes more complex. As stated above, this distribution problem is still much smaller than the distribution problem for traditional public keys where each entity has its own public key which needs to be distributed. Of course, traditional PKI technology can be used to distribute the public parameters, but a more suitable solution is to integrate the public parameters into the DNS lookup messages. In this way, the fact that a DNS lookup is made anyway to resolve a host IP is utilized, and the public parameter transfer can be piggybacked to the DNS reply. The technical details of the integration of IBC public parameter information into DNS records were evaluated by Smetters and Durfee [27]. Their positive evaluation lead us to adopt the public parameter distribution technique for our system. For more information on the details of how to incorporate this kind of information into the DNS system, the reader is referred to [27], [1] or [13]. To secure the transport, either DNSsec can be used or the public parameters can be signed and transferred with standard DNS, or a key agreement can be executed between the requesting party and the DNS server if the public parameters of the DNS server are known. Since the DNS server is usually in the same AS as the requesting customer, this is not a problematic issue, because the public parameters are the same as the customer's public parameters. As stated above, this part of the system has been tried and validated by several research groups.

4.2 Distribution of the Identity Keys

The most critical element in all IBEs or PKIs in key escrow mode is the distribution of identity keys (private keys) and the prevention of identity misbinding. In traditional PKI and IBE systems, this is usually done manually and out-of-band and thus creates a lot of work. While it can be argued that due to the fact that on the AS level most customers receive an out-of-band message when they receive their endpoint address, adding a fingerprint to the identity key would not put much extra burden on the system. However, a more elegant solution for the long term is to integrate the key distribution into the IP distribution system. For most networks, this means integration into the DHCP server. This, however, is not trivial since DHCP on its own is an unsecured protocol not suitable for transferring private information. The two main threats are packet sniffing and MAC spoofing. If the identity key is sent in the clear via the DHCP protocol in an unswitched network, an attacker can sniff the identity key, leading to key compromise. With MAC

spoofing, an attacker pretends to be the legitimate owner of a foreign MAC address, and the DHCP server sends the identity key to the attacker. Both forms of attacks make the plain use of DHCP for key distribution infeasible.

In the following, we present several solutions geared towards different scenarios of how the distribution of identity keys can be integrated into DCHP securely. In a fixed corporate network environment using a switched infrastructure, the easiest solution is to use the MAC lockdown function of modern switches. Using MAC lockdown, each port gets a MAC address and will only serve that MAC address. Thus, if an attacker wishes to spoof a MAC address to gain the key, physical access to the correct port must be acquired, significantly increasing the risk and effort of the attack. This scenario works in a corporate network where each MAC address is registered and assigned to a port anyway. In a student dormitory, for example, it is less feasible since managing the ever changing MAC addresses of the private devices used by students would be very time consuming and error prone. Here, an IEEE 802.1X + Radius [11,22] solution is more practical. The authorization is usually done in the form of a user-name password check. The IP address and the corresponding identity key can either be fixed (as set by the Radius and DHCP server) or dynamic and transient (more on transient IP addresses in section 4.3 and 4.4). Either way, only the legitimate user receives the identity key, and it is not possible to spoof the MAC address to receive a copy in the same key lifetime. If packet sniffing is an issue, the DHCP request needs to be extended to include a protected session key with which the identity can be protected from sniffing attacks. The client creates a session key which is encrypted using the public parameter N (N can be used in the same way as an RSA public key) of the key generator of the DCHP server and broadcasts the DHCP request. The session key can only be decrypted by the DHCP server which then uses the session key to encrypt the identity key of the client, using e.g. the Advanced Encryption Standard AES, which is then broadcasted. Thus, the identity key can only be decrypted by the client.

Apart from these two practical solutions based on an extension of existing security mechanisms which can be used in the short term, we also present a more speculative long term solution which does not rely on other security mechanisms. In this case, we bootstrap the network layer key agreement scheme on the data link layer by using MAC addresses as public keys. As with IP addresses, we cannot assume that there will be a single authority to generate the MAC identity keys, but since our system does not require cooperation between the ID-PKGs, this can be handled. Each organization with the authority to distribute MAC addresses runs its own ID-PKG and writes the identity key onto the networking card at the same time as the MAC address. Since the MAC addresses are globally unique and should not change over the lifetime of the networking card, a fixed identity key is not a problem. On the contrary, a hardware based protection of the key creates an added layer of security. Organizations with the right to distribute MAC addresses have their own Organizationally Unique Identifier (OUI) which is encoded in the first three octets of all MAC addresses distributed by this organization. Using this OUI, the public parameters needed for the MAC address can be found. This entails a very small and lightweight public parameter lookup mechanism matching OUIs to public parameters. This is the only step where some form of cooperation is needed on the organizational level, since all OUIs must be publicly available.

However, since the number of OUIs is small and does not change frequently, it is easy to solve this part of the distribution. The huge benefit of this structure is that the identity key distribution can now be automated in-band in a secure fashion without relying on extensive existing security mechanisms. Using this approach, it is possible for the requesting entity to add a proof of legitimate MAC address possession using the identity key of the MAC address when requesting its IP address. This not only prevents the problem of MAC spoofing, but also allows the DCHP server to send the identity key for the IP address to the requesting entity protected with the MAC based identity encryption. Since this mechanism is only used for requesting the identity key, which is done in an Intranet, the proposed solution does not open a backdoor to the Network Interface Card producers to decrypt the Internet traffic.

4.3 Key Expiration

Another practical issue of network layer encryption is the fact that especially in IPv4 networks, IP addresses are reused. In a PKI or CA based IPsec solution, this creates several problems, since the central PKI must be updated or the CA must be contacted to resign public keys as the users swap IP addresses. Certificate Revocation Lists can be used to accomplish this, but the response time until a change is propagated is quite long and creates a fair amount of effort. In particular, public key caching mechanisms can lead to problems. In our identity-based solution, natural key expiration techniques can be used to cope with dynamic IP addresses. Boneh et al. [6] showed how keys can be given a lifetime, which allows natural expiration of the identity key. This is done by the concatenation of the ID, in our case the IP address, with a date. The same technique can be used in our solution. In the scenario where ISPs have a pool of IP addresses which are allocated to customers on demand and reused at will, this technique can be used such that no two customers ever receive the same identity key. Since IP address reuse is time-delayed in any case[1], this time frame can be used as the key lifetime to ensure that each successive owner lies in a new lifetime slot. With the techniques introduced in this paper, a frequent automatic in-band key distribution can be safely executed and thus key renewal is far less of a problem. Additionally, key expiration also reduces the risk of identity key theft, since the attack window is restricted to a small time interval.

4.4 Transient Key Generation

If a large network requires the uses of transient keys, the key generator can be required to generate a large number of identity keys. To ease the computational load of the key generator, we implemented an extension to the generation protocol which makes the generation of identity keys less computationally expensive, at the cost of making the session key calculation more expensive. This distributes the load from a single point (the key generator) to a large number of resources (the clients). The extension makes uses of the fact that exponentiation using repeated squaring is faster when the binary representation of the exponent is a sparse-one integer. A sparse-one integer D of order

[1] Before an IP address is allocated to a new user, a certain amount of time must pass to prevent attackers from impersonating the previous entity.

δ is an integer whose binary representation has exactly δ 1's. The extended algorithm is presented below.

Extract - The **Extract** algorithm is executed by the ID-PKG.
Input: PSP, SP, ID
Let ID be a given identity. The algorithm computes $d_{ID} \equiv H(ID)^D \ mod \ N$.
Choose D as a sparse-one integer. The PSP are increased by the value E from $D \cdot R \equiv E \ mod \ \varphi(N)$.
Output: d_{ID}

There are exactly $\binom{k}{\delta}$ sparse-one integers of order δ with bit-length k. Thus, δ should be selected to be sufficiently large to make exhaustive search infeasible.

4.5 NAT Traversal

The final practical issue is the NAT problem. While this mainly is a problem in IPv4 networks, there are also scenarios in IPv6 networks in which NAT is an issue. The main problem when dealing with network layer encryption when NAT is involved is that the NAT server substitutes its public IP address for the private IP address of the entity being NATed. As such, the original identity key for the private IP address is no longer valid, since it does not match the public IP address of the NAT router, hence any key agreement would fail. This problem is also faced by IPsec which has problems with NATed resources. When working in a NAT environment, a certain level of trust must exist between the NAT router and the NATed device. The NAT router substitutes its public IP address for the private IP address of the NATed device. The NATed device must trust the NAT router to substitute the right address, and the NAT router must be willing to forward the packets on behalf of the NATed device. However, when using encryption, the NATed device does not trust the NAT router with the plain text version of its communication. Communication between the NATed device and the outside world should still be private. Considering that the NAT router shares its public IP address with the NATed devices, our solution also lets the NAT router share the identity key of its public IP address with the NATed devices (we will show later that this does not compromise the security of either the NAT router or the NATed devices). The identity key of its Intranet IP address is, however, kept private. Also, a private identity key is given to each NATed device, corresponding to its Intranet IP address. When a NATed device A in the Intranet establishes a connection to an external device B, it creates a SIK packet using its private value G^{r_A} in combination with the identity key of the NAT router's public IP address. This is, in essence, an extension of the normal NAT procedure to include an authenticated key exchange, and the trust relationship between the NAT router and the NATed device is not changed. The sharing of an identity key belonging to an IP address is not usual and should be avoided under normal circumstances, since anyone in possession of the identity key can pose as the legitimate owner of the corresponding IP address and thus can spoof the address or act as a man-in-the-middle attacker. However, in the NAT scenario this is exactly the desired outcome, since the NATed devices pretend to be the NAT router to the outside world, since as far as the outside world is

concerned, the packets originate from the NAT router. It is important to note that although the identity key of the NAT routers' public IP address is used by the NATed device, the NAT router is not able to subvert the communication. To successfully attack the communication as a man-in-the-middle, the NAT router would also need to be in the possession of the private identity key of *B*, which is not the case. It is also not critical if more than one device is behind the same NAT router, since communication between the NATed devices and the NAT router is protected by the private identity key of the NAT router's Intranet IP address and the identity key of the NATed device, which is different for each device. Thus, the NATed devices are not able to subvert the communication of other devices in the Intranet nor are they able to spoof the internal identity of the NAT router or other NATed devices. Should the Intranet devices be connected to the NAT router with a pre-configured switch, the Intranet identity keys are not necessary, since the private value G^{r_A} of the key agreement is sufficient to protect the key exchange if there is a direct connection to the NAT router.

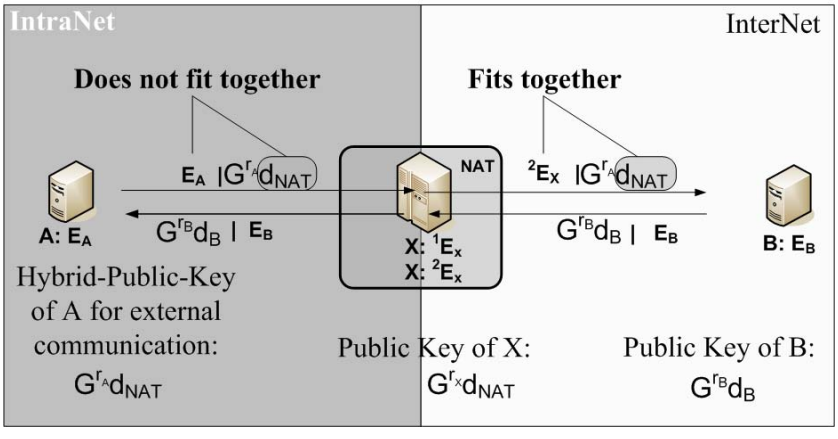


Fig. 1. Proposed solution for the NAT problem with endpoint keys

Figure 1 shows the solution for the NAT problem. The internal user *A* sends a SIK using its own private value G^{r_A} in combination with the private key of the NAT router's IP address. When the NAT router substitutes the IP address with its own, it creates a valid packet, since the value d_{NAT} now belongs to the correct source address of the packet.

5 Conclusions

In this paper, a new identity-based key agreement protocol has been presented to secure the communication channel between two devices using their endpoint addresses as public keys for authentication. The proposed approach has been demonstrated using IP addresses on the network layer and MAC addresses on the data link layer to bootstrap the system, which has allowed us to piggyback much of the security overhead for

key management to the existing network infrastructure. Unlike other identity-based encryption solutions, the presented approach is based on the well tested mathematics also used in the traditional Diffie-Hellman key agreement and Rivest-Shamir-Adleman public key cryptography approaches instead of elliptic curves or quadratic residues. It has been shown how our identity-based key agreement protocol can be used as a generic low level security mechanism and how it can deal with the implementation issues of multi-domain key generation, key distribution, multi-domain public parameter distribution, key expiration, inter-domain key agreement and network address translation traversal.

There are several areas of future work. For example, a more detailed description of the integration of the proposed identity-based approach into existing network management protocols and tools, in particular the integration into the DHCP protocol, should be provided. Furthermore, the large-scale practical deployment of the proposed approach in IP, Voice-over-IP, or mobile telephone communication scenarios is an interesting area for future work.

References

1. Adida, B., Chau, D., Hohenberger, S., Rivest, R.L.: Lightweight Email Signatures (Extended Abstract). In: 5th International Conference on Security and Cryptography for Networks, pp. 288–302 (2006)
2. Appenzeller, G., Lynn, B.: Minimal-Overhead IP Security Using Identity Based Encryption, Technical Report, Voltage Inc. (2002)
3. Schridde, C., Smith, M., Freisleben, B.: An Identity-Based Key Agreement and Signature Protocol with Independent Private Key Generators. Technical Report, Dept. of Mathematics and Computer Science, University of Marburg, Germany (2008)
4. Aura, T.: Cryptographically Generated Addresses, RFC 3972 (2005)
5. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
6. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. SIAM Journal of Computation 32(3), 586–615 (2003)
7. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
8. Chen, L., Cheng, Z., Smart, N.P.: Identity-based Key Agreement Protocols from Pairings. International Journal of Information Security 6(4), 213–241 (2007)
9. Chen, L., Kudla, C.: Identity Based Authenticated Key Agreement Protocols from Pairings. In: 16th IEEE Computer Security Foundations Workshop (CSFW 2003), p. 219 (2003)
10. Cocks, C.: An Identity Based Encryption Scheme Based on Quadratic Residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)
11. Congdon, P., Aboba, B., Smith, A., Zorn, G., Roese, J.: IEEE 802.1X Remote Authentication Dial. In: User Service (RADIUS) Usage Guidelines, RFC 3580 (September 2003)
12. Diffie, W., Hellman, M.E.: New Directions In Cryptography. IEEE Transactions On Information Theory (6), 644–654 (1976)
13. Fenton, J., Allman, E., Libbey, M., Thomas, M., Delany, M., Callas, J.: DomainKeys Identified Mail (DKIM) Signatures, RFC 4870 (2007)

14. Günther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
15. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
16. Maurer, U.M., Yacobi, Y.: A Non-Interactive Public-Key Distribution System. Designs, Codes and Cryptography 9(3), 305–316 (1996)
17. McCullagh, N., Barreto, P.: A New Two-Party Identity-Based Authenticated Key Agreement. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 262–274. Springer, Heidelberg (2005)
18. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Host Identity Protocol, RFC 4423 (October 2003)
19. Okamoto, E.: Key Distribution Systems Based on Identification Information. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 194–202. Springer, Heidelberg (1988)
20. Pohlig, S.C., Hellman, M.E.: An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance. IEEE Trans.on Info. Theory IT-24, 106–110 (1984)
21. Pollard, J.: Theorems of Factorization and Primality Testing. Mathematical Proceedings of the Cambridge Philosophical Society 76, 521–528 (1974)
22. Rigney, C., Rubens, A., Simpson, W., Willens, S.: Remote Authentication Dial In User Service (RADIUS), RFC 2138 (April 1997)
23. Rivest, R.L., Shamir, A., Adleman, L.: A Method For Obtaining Digital Signatures And Public-Key Cryptosystems. Communications Of ACM 1(2), 120–126 (1978)
24. Sakai, R., Kasahara, M.: ID based Cryptosystems with Pairing on Elliptic Curve. In: Symposium on Cryptography and Information Security (2003)
25. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
26. Smart, N.P.: Identity-based Authenticated Key Agreement Protocol based on Weil Pairing. Electronics Letters 38(13), 630–632 (2002)
27. Smetters, D.K., Durfee, G.: Domain-based Administration of Identity-Based Cryptosystems for Secure E-Mail and IPSEC. In: SSYM 2003: Proceedings of the 12th Conference on USENIX Security Symposium, Berkeley, CA, USA, p. 15. USENIX Association (2003)

Author Index