Advances in
COMPUTERS

Information Security

EDITED BY

MARVIN V. ZELKOWITZ

Department of Computer Science
and Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland

VOLUME 60

ELSEVIER
ACADEMIC
PRESS

Amsterdam Boston Heidelberg London New York Oxford
Paris San Diego San Francisco Singapore Sydney Tokyo

This page intentionally left blank

Advances

in COMPUTERS
VOLUME 60

This page intentionally left blank

ELSEVIER B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam
The Netherlands

ELSEVIER Inc.
525 B Street, Suite 1900
San Diego, CA 92101-4495
USA

ELSEVIER Ltd
The Boulevard, Langford Lane
Kidlington, Oxford OX5 1GB, UK

ELSEVIER Ltd
84 Theobalds Road
London WC1X 8RR, UK

First edition 2004

Library of Congress Cataloging in Publication Data
A catalog record is available from the Library of Congress.

British Library Cataloguing in Publication Data
A catalogue record is available from the British Library.

ISBN: 0-12-012160-3
ISSN (Series): 0065-2458

♾ The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).

Printed in Great Britain.

Contents

The Digital Detective: An Introduction to Digital Forensics

Warren Harrison

Survivability: Synergizing Security and Reliability

Crispin Cowan

Smart Cards

Katherine M. Shelfer, Chris Corum, J. Drew Procaccino, and Joseph Didier

Shotgun Sequence Assembly

Mihai Pop

Advances in Large Vocabulary Continuous Speech Recognition

Geoffrey Zweig and Michael Picheny

This page intentionally left blank

Contributors

Donald J. Bagert is a Professor of Computer Science and Software Engineering at the Rose-Hulman Institute of Technology, where he also has the title of Director of Software Engineering. Don came to Rose-Hulman in August 2002 after 14 years at Texas Tech University. His current research interests include software process improvement, software tools for student advising, and software methodologies. Don received a B.S. in Engineering from Tulane University, an M.S. in Computer Science from the University of Louisiana at Lafayette in 1979, and a Ph.D. in Computer Science from Texas A&M University in 1986. He is the first person licensed as a Professional Engineer in software engineering in both Texas and the United States. In 2002–03, Don was among the first group of ABET software engineering program evaluators. He is also the Managing Editor of *FASE*, an electronic newsletter devoted to software engineering education, training and professional issues. Don is also the chairperson of the CSDP Certification Committee, a member of both the Educational Activities Board and the Professional Practices Committee for the IEEE Computer Society, and is a Senior Member of the IEEE.

Chris Corum, co-founder of AVISIAN Inc., serves as Editor-in-Chief of the ContactlessNews, CR80News, and SecureIDNews. In his work with AVISIAN, he provides ID technology guidance to government agencies including the Department of Defense, leading financial institutions, and a range of corporate security entities. Previously he served as Director of Marketing for CyberMark, a pioneering smart card systems integrator, and as a smart card and biometric technology analyst for Florida's Legislative Information Technology Committee. Mr. Corum's degrees are in Advertising, Information Technology Marketing, and he is currently completing a Ph.D. in Communication.

Crispin Cowan is co-founder and Chief Scientist of Immunix, Inc. His research interests include making systems more secure without breaking compatibility or compromising performance. He has coauthored 33 refereed publications, including those describing the StackGuard compiler for defending against buffer overflow attacks. He has a Ph.D. in computer science from the University of Western Ontario, and Masters and Bachelors degrees in computer science from the University of Waterloo. Contact him at Immunix, 920 SW 3rd Ave., Ste. 100, Portland, OR 97204.

George Cybenko is the Dorothy and Walter Gramm Professor of engineering at Dartmouth College. He received his B.Sc. in mathematics at the University of Toronto, and an M.A. in mathematics and Ph.D. in electrical engineering/computer science and applied mathematics from Princeton. He has taught on the computer science faculty at Tufts University and was professor of electrical engineering and computer science at the University of Illinois, Champaign–Urbana. At Illinois, he was also a director of the university's Center for Supercomputing Research and Development. He has served as editor for seven mathematics, computer, and information theory publications, has helped organize a dozen conferences and symposia, and has published over fifty journal papers, book chapters, and conference proceedings. He has also delivered over 100 research lectures at universities, symposia, and colloquia around the world.

Joseph W. Didier is President & CEO of InfinaCard Inc. Joseph has a wide-range of knowledge and experiences in card technology, smart cards, stored value, financial payment, transaction processing, and loyalty programs. Prior to launching his own company, he was a Sales Executive for ACI Worldwide where he provided enterprise payment solutions; Director of Marketing for a start-up smart card solutions provider; and he was directly involved with the development of the first multi-applicational smart card system at The Florida State University. Joseph holds dual Bachelor of Sciences degrees from The Florida State University in Entrepreneurship & Small Business Management and Business Management. He has a Master in Business Administration degree from Drexel University.

Annarita Giani is pursuing a Ph.D. in computer engineering at the Institute for Security Technology Studies and Thayer School of Engineering at Dartmouth College. Her research interests include communication theory, signal analysis, and computer security. She received a Laurea in applied mathematics from the University of Pisa.

Warren Harrison is Professor of Computer Science at Portland State University and a Police Reserve Specialist with the Hillsboro (Oregon) Police Department. He received his Ph.D. in Computer Science from Oregon State University and currently serves as Editor-in-Chief of IEEE Software Magazine. His research interests include software engineering, digital forensics and mobile Internet technology.

Mihai Pop is a bioinformatics scientist at The Institute for Genomic Research (TIGR) in Rockville, Maryland. His research interests include sequence assembly algorithms and their practical applications as well as sequencing and finishing techniques. He is the author of the open-source scaffolder Bambus, and an active member of a collaborative effort to create a modular open-source assembler (AMOS). Mihai Pop holds a Ph.D. in computer science from the Johns Hopkins University in Baltimore, Maryland.

Michael Picheny is the Manager of the Speech and Language Algorithms Group in the Human Language Technologies Group at the IBM TJ Watson Research Center. Michael has worked in the Speech Recognition area since 1981, joining IBM after finishing his doctorate at MIT. He has been heavily involved in the development of almost all of IBM's recognition systems, ranging from the world's first real-time large vocabulary discrete system through IBM's current ViaVoice product line. Michael served as an Associate Editor of the IEEE Transactions on Acoustics, Speech, and Signal Processing from 1986–1989, is currently a member of the Speech Technical Committee of the IEEE Signal Processing Society and its representative to the Signal Processing Society conference board, and is a Fellow of the IEEE.

J. Drew Procaccino is an Assistant Professor of Computer Information Systems in the College of Business Administration at Rider University (Lawrenceville, NJ). His teaching interests include office productivity software, systems analysis and design, systems development and database design. His research interests include electronic commerce, software engineering, biometrics and smart card technology.

Katherine M. Shelfer is Assistant Professor of Competitive Intelligence, College of Information Science and Technology, Drexel University in Philadelphia, PA, where she also directs the Competitive Intelligence Certificate Program. Her research and teaching interests include competitive intelligence, sources of business information and the strategic implications of information systems and services, specifically smart card systems. Dr. Shelfer has published work on smart cards in Communications of the ACM, the Defense Intelligence Journal and Knowledge Management for the Information Professional, among others.

Paul Thompson is senior research engineer at the Institute for Security Technology Studies and Thayer School of Engineering at Dartmouth College. His research interests include document retrieval, information extraction, and computer security. He received a Ph.D. in library and information studies from the University of California, Berkeley.

Geoffrey Zweig is the Manager of Advanced LVCSR Research at the IBM TJ Watson Research Center. He received a B.A. degree in Physics with highest honors in 1985, and a Ph.D. in Computer Science in 1998, from The University of California at Berkeley. Following his thesis on the application of Bayesian Networks to ASR, Geoffrey joined IBM in 1998. In 2001, he co-authored the Graphical Models Toolkit for speech recognition. He participated in the 2001 DARPA-sponsored HUB-5 evaluations, and again in the 2003 EARS Rich Transcription evaluation. Geoffrey is currently a member of the IEEE, and Associate Editor of IEEE Transactions on Speech and Audio Processing. His research interests include the development of multi-scale acoustic models, the application of machine learning techniques to speech recognition, and reliable, efficient decoding techniques.

This page intentionally left blank

Preface

Advances in Computers is the oldest series to provide an annual update to the continuously changing information technology field. It has been continually published since 1960. Within each volume are usually six to eight chapters describing new developments in software, hardware, or uses of computers. In this 60th volume of the series, subtitled *Information Security*, the focus of most of the chapters is on changes to the information technology landscape involving security issues. With the growing ubiquity of the Internet and its growing importance in the everyday life, the need to address computer security issues is growing. The first 5 chapters describe aspect of this information security problem. The final two chapters present other topics of great interest and importance today—genome sequencing and speech recognition.

In Chapter 1, "Licensing and certification of software professionals," Professor Donald J. Bagert discusses the current controversy of certifying software professionals. Should software engineers be licensed? What does that mean? What is the body of knowledge that defines what a software professional should know? Should educational programs be accredited like most engineering programs? All of these are hotly debated today, and given the impact that computer software has on the world's economy, some resolution to these issues must be forthcoming.

Any user of computers today should understand the danger that viruses, worms, and Trojan horses have on the integrity of their computer system. Most attacks are known after they occur and after the damage has already been done. But in Chapter 2, "Cognitive Hacking" by George Cybenko, Annarita Giani, and Paul Thompson, the authors discuss a different form of attack where neither hardware nor software is necessarily corrupted. Rather the computer system is used to influence people's perceptions and behavior through misinformation. For example, anyone can post anything on a Web page with few limitations. The issue is how to deal with false information on the Web and how to decide whether a source is reliable. This and related topics are the focus of this chapter.

Most people, criminals included, store information on computers. After a crime has been committed and a suspect arrested, how do you provide evidence in a court of law that an illegal action did occur and that the suspect was indeed responsible? This is the domain of computer forensics. In Chapter 3, Warren Harrison discusses

"The digital detective: An introduction to digital forensics." It is estimated that half of all federal criminal cases require a computer forensics examination. This chapter addresses the identification, extraction and presentation of evidence from electronic media as it is typically performed within law enforcement agencies.

In Chapter 4 "Survivability: Synergizing security and reliability" by Crispin Cowan, the author discusses the issue of survivability of a computer system. The goal of the chapter is to show how security and reliability techniques are required to achieve survivability, the ability of a system to continue to operate in the face of failures. In the context of computer security, survivability is the study of how to mask security faults, and do so such that attackers cannot bypass the fault masking.

Chapter 5 "Smartcards" by Katherine M. Shelfer, Chris Corum, J. Drew Procaccino, and Joseph Didier, is the final chapter in this section on information security. Credit cards are now ubiquitous and vital to the economic well being of most national economies. Because of their widespread use, there is interest in providing additional information on such cards. By adding a processor directly on the card itself, additional security, as well as additional functionality, can be provided to both the user and merchant. This chapter discusses the development of these "smart" cards.

Chapter 6 "Shotgun sequence assembly" by Mihai Pop discusses the important issue of genome sequencing. With the decoding of human DNA into sequences, biology, especially medical research, has been greatly transformed in the past 10 years. In this chapter, Dr. Pop discusses the shotgun sequencing technique used to decipher the complete genome of various bacteria and viruses.

The final chapter, "Advances in large vocabulary continuous speech recognition" by Geoffrey Zweig and Michael Picheny discusses the advances in accurate and efficient speech recognition systems. These are becoming quite common for customer service, broadcast news transcription and automated directory assistance, among other commercial applications. This chapter discusses the underlying technology that is behind the increasing success of these systems.

I hope that you find these articles of interest. If you have any suggestions of topics for future chapters, or if you wish to contribute such a chapter, I can be reached at mvz@cs.umd.edu.

Marvin Zelkowitz
University of Maryland,
College Park, MD, USA

Licensing and Certification of Software Professionals

DONALD J. BAGERT

Rose-Hulman Institute of Technology
5500 Wabash Avenue, CM97
Terre Haute, IN, 47803-3999
USA
Don.Bagert@rose-hulman.edu

Abstract

For many years, software organizations have needed to hire developers with a wide range of academic and professional qualifications, due to the ongoing shortage of individuals qualified to create and maintain the products required to satisfy marketplace demand. Many of these companies have used the certification credentials of such individuals to help judge whether they have the proper background for the development requirements of their particular software organization. *Certification* is a voluntary process intended to document the achievement of some level of skill or capability. Such certification can be awarded through a variety of organizations. To date, company-based certification programs have been dominant in the software field. These programs have been created and run by a particular company, and are usually centered on determining an individual's qualification to use a particular type of software that is marketed by that business. However, these programs are often limited in scope, and sometimes make it possible to acquire certification with little practical software development background or formal training.

However, there have recently been a growing number of efforts to provide more comprehensive certification programs for software professionals through professional societies and independent organizations. Some of such certificates are offered as a specialization in areas that in a number of fields are a part of the product development process, e.g., quality assurance and project management. In other cases, there are programs intended to certify individuals for having general knowledge and abilities across a wide range of software development areas. In some countries, such certification of software engineering professionals is done on a nationwide basis by an engineering professional society.

There has also been an increased interest in the *licensing* of software engineering professionals. Licensing is a more formal version of certification that

ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60001-X

involves a government-sanctioned or government-specified process, with the health, safety and welfare of the public in mind. Since engineering is a field where licensing is commonplace in many countries, most of this effort has focused on the licensing of software engineers. However, while licensing is commonplace in professions such as law and medicine, it has until recently been virtually unknown in the information technology field. A number of IT professionals have raised a variety of concerns about the licensing of software engineers, including issues related to liability and the body of knowledge upon which to base such licensing programs.

This chapter will examine the various licensing and certification initiatives, including the history of its development, the process and content of such programs, and the arguments both for and against licensing and certification.

1. Introduction

1.1 Overview

The number of software professionals in the workforce is large and growing. In the United States, a December 2001 study by the Department of Labor [39] stated that the number of people employed by software engineers in the United States was

697,000, and it was expected that there would be 1.36 million software engineering jobs available in the U.S. by 2010. (These numbers are in addition to computer programmers and other information technology-related positions, which number over 2.2 million in 2000, and is projected to grow to over 3.5 million by 2010.) In addition, these workers come from a wide variety of educational and work experiences. Since there is such a large and diverse pool of potential workers, many employers will look to see if job applicants possess any professional certifications or licenses as one factor in evaluating their credentials.

Certification is a voluntary process intended to document the achievement of some level of skill or capability. This type of certification can be given through a variety of organizations, such companies, professional societies, or institutes who primary function is to award such credentials. To date, *company-based* certification programs have been dominant in the software field. Such programs have been created and run by a particular company (such as Microsoft), and are usually centered on determining an individual's qualification to write programs or otherwise use software that is marketed by that business. Therefore, company-based certification is by definition usually limited in scope, and in some cases is possible to acquire with little practical software development experience and no formal education. The Microsoft Certified Solution Developer (MCSD) program is the one that is most related to software engineering offered by Microsoft, and so will be discussed in more detail later in this article.

However, due to the aforementioned large and growing size of the software engineering community, there have recently been a growing number of efforts to provide more comprehensive certification programs for software professionals through professional societies and independent organizations. Some such *institute-based* certifications are offered as a specialization in areas that in a number of fields are a part of the product development process, e.g., the Software Quality Engineer certification provided by the American Society for Quality (ASQ); others, such the IEEE Certified Software Development Professional (CSDP) program, are intended to certify individuals for having general knowledge and abilities across a wide range of software development areas.

There has also been an increased interest in the *licensing* of software engineering professionals, especially in North America. Licensing is a more formal version of certification that involves a government-sanctioned or government-specified process, with the health, safety and welfare of the public in mind. Since engineering is a field where licensing is commonplace in many countries, most of this effort has focused on the licensing of software engineers. However, while licensing is commonplace is professions such as law and medicine, until recently it has been virtually unknown in the information technology (IT) field. A number of IT professionals have raised a variety of concerns about the licensing of software engineers, including issues re-

lated to liability, the body of knowledge upon which to base examinations, and the appropriateness of the engineering model for such licensing.

Some countries either implicitly or explicitly designate that nation's primary engineering professional society to certify engineers. Such *national certification* of professional engineers is done through a process similar to licensing in the United States, except that national certification (as the name implies) is a voluntary process.

This article will examine the various licensing and certification initiatives, including the history of its development, the process and content of such programs, and the arguments both for and against licensing and certification. The remainder of Section 1 looks how to view the area of competency for which someone is being certified or licensed, and provides an outline of the steps commonly required in a certification or licensing process. Section 2 will examine licensing and national certification, Section 3 looks at institute-based and company-based certification; and the final section will make some conclusions and outline some possible future directions for the licensing and certification of software professionals.

1.2 Area of Competency

A competency area for professional certification and licensing is likely to include (directly or indirectly) the following components:

- Body of knowledge
- Education and training
- Code of ethics and professional conduct

As the name implies, the *body of knowledge* (BOK) of a particular subject encompasses the information essential for practitioners in that area. In order to be certified or licensed in a particular subject or field, a software professional would need to demonstrate a particular level of understanding of the appropriate BOK. For a company-based certification program, this would likely involve understanding how to program using a particular set of company's application software. In the case of the licensing of software engineers, knowledge in a much wider range of subjects is required. An extensive project that has compiled a guide to the body of knowledge of software engineering is discussed in Section 2.2.

In order to obtain the knowledge necessary to demonstrate competency, some type of *education and training* is almost always required. For licensing, this would include a baccalaureate degree in a discipline related to software engineering, possibly from an accredited program (see Section 2.3). The administrators of most certification programs provide training materials or seminars to help prepare potential applicants. (As of July 2003, Microsoft had 11,500 certified trainers for their programs.) Some

high schools and two-year colleges in the United States offer courses to help students prepare for particular company-based certification exams [6].

Most professions also have a *code of ethics and professional conduct* as defined by either a related professional society or a legal authority. The two major U.S. computing professional societies have developed such a code for software engineering (see Section 2.1). Most professional licensing jurisdictions will specify a code of ethics and professional conduct required of all licensees. An organization which manages a certification program usually has a code of ethics which applicants agree to adhere to as part of their certification.

1.3 Procedure

One or more of the following pieces of information is commonly used in the assessment of an application for certification or a license:

- Educational background
- Work experience and professional references
- Examinations passed

As previously stated, most professional licenses require a related baccalaureate degree. Some certifications (such as CSDP) require a particular formal educational background, while others (such as the Microsoft certifications) do not.

Most engineering license boards require four or more years of engineering experience, preferably under the supervision of a licensed engineer. They also require references from professional engineers who have had an opportunity to observe the applicant's work. The CSDP requires 9000 hours of software development experience, but no accompanying references. The Microsoft certification programs recommend some practical experience before attempting their examinations, but do not require it (or any references).

Most certification programs have a person apply before administering any examinations. In the U.S., the application for a state engineering license is submitted after the applicant has passed two-nationally administered examinations; however, In Texas, however, there is a rule which allows the waiver of such exams with additional experiences and references (see Section 2.5). A test on that state's engineering practice laws is also usually required for licensure.

There is also an application fee, and there may be separate examination fees.

1.4 Renewal

Most certification programs are for a limited time, and require some type of recertification mechanism. Some licensing boards only require the payment of annual

fees, but an increasing number require of ongoing continuing education. Microsoft certifications are based on a set of subject examinations, and if a new examination comes out a particular subject, certificate holders are required to recertify by taking the new exam within a certain time period. (Also, most Microsoft examinations are eventually discontinued, as particular platforms are no longer supported, thus requiring the certificate holder to be re-examined for the new operating system.)

1.5 Summary

Table I summarizes the different aspects of licensing, and the different types of certification.

2. Licensing of Software Engineers

2.1 The Nature and Development of Software Engineering

The term *software engineering* has been in use since the late 1960s, and has been for many years in common use by the general public. This is the despite the fact software engineering does not fit the standard profile of an engineering field for several reasons, including:

- Software is a non-physical product, and therefore acts upon a different set of scientific principles (mostly involving computer science) than are used in other engineering disciplines, which base their principles on other engineering sciences such statics, thermodynamics, and strength of materials;
- While other engineering disciplines focus on design, a software engineer could potentially work in any aspect of the process life cycle, thus also acting in the roles traditionally held by architects, construction workers, and maintenance personnel, among others; and
- Software is developed in a wide range of application domains, encompassing virtually all aspects of modern life.
- Software professionals have a wide variety of educational background and experience; for instance, one might be a high school dropout with several years of professional programming work, while another has a Bachelor of Science in Software Engineering and no work experience.

Despite this, there are also strong arguments for identifying software development as an engineering field, including the use of a process very similar to that used in “traditional” engineering disciplines, with the same goals in mind: to efficiently develop a reliable and usable product, on time and within budget estimates.

TABLE I
A COMPARISON OF LICENSING AND CERTIFICATION FEATURES

	Licensing	**National Certification**	**Institute-Based Certification**	**Company-Based Certification**
Granting organization	Government-based or government-sanctioned body	That nation's primary society for that profession	Private institute or professional society	Company which developed the software being used
Legal permission to practice granted?	Yes	No	No	No
Body of knowledge involved	For entire professional discipline of license field	For entire professional discipline of certification field	For certification area	For certification area
Formal Education	Usually a baccalaureate degree in related field	Usually a baccalaureate degree in related field	None usually required, but may reduce experience requirement	Usually none required
Code of ethics and professional conduct	Adherence required	Adherence required	Adherence usually required	Usually none are directly involved
Work experience	Usually four years or more in discipline	Usually four years or more in discipline	Usually required; amount may vary depending on the amount of formal education	Usually none required
Professional references	Usually required	Usually required	Usually not required	Usually not required
Passage of examinations required?	Varies	Varies	Yes	Yes

At any rate, since "software engineering" is firmly entrenched in the lexicon, the ramifications of such terminology need to be addressed. At first, software engineering was considered a specialization; however, in the last twenty years it has been increasingly regarded as a separate discipline and profession. This is turn has potential ramifications for the licensing and certification. Frailey [23] asserts that four facts need to be established in order to determine that licensing or certification of software engineers:

1. That software engineering is a separate discipline,
2. That software engineering is a profession,
3. That software engineering is sufficiently established to justify certification or licensing, and
4. That certification or licensing of software engineering would be beneficial enough to justify the effort to establish them.

In the United Kingdom, the engineering and computing communities came to the conclusion over a decade ago that these facts were indeed established, and thus began creating undergraduate software engineering degree programs, and bestowing Chartered Engineer status to qualified individuals in the field. However, in the United States and other countries, the process has been somewhat slower, and even today there are many in the global engineering and computing communities that feel that software engineering is not a separate discipline, and even if it is, is not engineering.

An important step came in 1993 with the creation of the *ad hoc* Joint IEEE Computer Society and ACM Steering Committee for the Establishment of Software Engineering as a Profession (http://www.computer.org/tab/seprof). Although the Association of Computing Machinery (ACM) and the Computer Society of the Institute of Electrical and Electronics Engineers, Inc. (IEEE-CS) are both based in the United States, they each have a significant international component, and consider themselves as representing the computing community worldwide.

The mission statement of the Joint Steering Committee was "To establish the appropriate sets(s) of criteria and norms for professional practice of software engineering upon which industrial decisions, professional certification, and educational curricula can be based". They established three task forces: Ethics and Professional Practices, Body of Knowledge and Recommended Practices, and Education. (Note that these correspond to the essential components for licensing and certification described in Section 1.2.)

Work by the ethics task force proceeded quickly, and *Software Engineering Code of Ethics and Professional Practice* was approved by both ACM and the Computer Society in 1999 [25]. The body of knowledge task force did a "pilot survey" of software engineers to gather some initial data concerning the body of knowledge [20]. It was apparent from the creation of this survey that a volunteer task force would

not have the resources required to properly compile the software engineering body of knowledge, which led to the creation of the Guide to the Software Engineering Body of Knowledge (SWEBOK) project in 1997. (SWEBOK will be discussed in detail in Section 2.2.)

Since the education task force's objective was to develop Software Engineering Education recommendations based on the work of the body of knowledge task force, most of their work would be delayed until there was further definition of the software engineering BOK. In the meantime, accreditation efforts went forward in Australia, Canada and the United States (see Section 2.3).

With the recognition that the *ad hoc* joint steering committee would need to be an ongoing effort, in late 1998 ACM and IEEE-CS voted to replace that group with the Software Engineering Coordinating Committee (SWEcc or SWECC), which would be a standing committee of the two societies (homepage http://computer.org/tab/swecc.htm). The mission of SWECC was similar to that of its predecessor, except that its structure would be different, in that instead of taking on projects itself, it would coordinate various software engineering-related projects approved and funded by the two societies. So, the new SWEBOK project reported to SWECC, as well as the Software Engineering Education Project (SWEEP), which was a continuation of the Education Task Force of the *ad hoc* committee.

All seemed to be progressing well—except that many (especially in ACM) were very concerned that all the elements required for the licensing of software engineers were coming into place, as will be discussed in Section 2.5. This eventually led to the effective dissolution of SWECC.

2.2 The Guide to the Software Engineering Body of Knowledge (SWEBOK)

2.2.1 Overview

As previously stated, due to the depth and breadth required to compile a body of knowledge for software engineering, it was determined that such a project required full-time rather than volunteer management. For this reason, IEEE-CS (and later SWECC) contracted the Software Engineering Management Research Laboratory at the Université du Québec à Montréal to manage the SWEBOK effort; the project was later moved to the École technologie supérieure (ÉTS) in Montreal. The SWEBOK website is at http://www.swebok.org. The project is still ongoing, although it is currently scheduled for completion at the end of 2003.

The SWEBOK project team established the project with five objectives:

1. Characterize the contents of the software engineering discipline.

2. Provide topical access to the software engineering body of knowledge.
3. Promote a consistent view of software engineering worldwide.
4. Clarify the place and set the boundary of software engineering with respect to other disciplines such as computer science, project management, computer engineering, and mathematics.
5. Provide a foundation for curriculum development and individual certification material [13].

That last objective is of the most interest to this article, and has perhaps been the most controversial aspect of the project.

It is important to note that the product of the SWEBOK project was not intended to be the body of knowledge itself, but rather a guide to it (thus its full name). A consensus on the core subset of knowledge characterizing the software engineering discipline was sought in this project.

The SWEBOK project has a number of sponsors, including the IEEE Computer Society, the (U.S.) National Institute of Standards and Technology, the National Research Council of Canada, Canadian Council of Professional Engineers, MITRE, Rational Software, SAP Labs-Canada, Construx Software, Raytheon, and Boeing, all of whom have representatives on the project's Industrial Advisory Board. The project's three-man Panel of Experts consists of well-known software engineering book authors Roger Pressman, Ian Sommerville and Steve McConnell.

The SWEBOK project also has a relationship to the normative literature of software engineering as embodied in software engineering standards and related documents. Version 1.0 of the SWEBOK Guide [14], released in May 2001, is considered an IEEE standard, and is in the final stages of receiving ISO (International Organization of Standards) approval. Three review cycles were conducted before Version 1.0 was released for a two years of trial use. After the trial period ended, another review cycle was conducted as a prelude to a revision of the document for Version 2.0, to be released at the end of 2003.

2.2.2 Body of Knowledge Content

In [13], the authors state that "From the outset, the question arose as to the depth of treatment the Guide should provide. After substantial discussion, we adopted a concept of *generally accepted* knowledge ... which we had to distinguish from advanced and research knowledge (on the grounds of maturity) and from specialized knowledge (on the grounds of generality of application). The generally accepted knowledge applies to most projects most of the time, and widespread consensus validates its value and effectiveness... However, generally accepted knowledge does not imply that we should apply the designated knowledge uniformly to all software engineering endeavors each project's needs determine that—but it does imply that

competent, capable software engineers should be equipped with this knowledge for potential application."

Furthermore, it was intended (through a recommendation by the SWEBOK Industrial Advisory Board) that this generally accepted knowledge would be appropriate in the study material for a software engineering licensing examination that graduates would take after gaining four years of work experience.

Version 1.0 of the SWEBOK Guide classifies the information compiled on software engineering using ten knowledge areas (KA), as shown in Table II. Each KA was authored by a leading expert in that particular area. The Guide also identifies seven disciplines related to (but not part of) software engineering, such as computer science and project management (Table III).

The resulting SWEBOK guide is one that reads somewhat like the very popular software engineering textbooks written by Pressman [34] and Sommerville [36], but has two major differences: it is intended for a different audience (practitioners rather than college students), and is the result of a more widespread review and consensus, as opposed to the vision of a single author.

TABLE II
SWEBOK KNOWLEDGE AREAS

Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Tools and Methods
Software Quality

TABLE III
SWEBOK RELATED DISCIPLINES

Computer Science
Mathematics
Project Management
Computer Engineering
Systems Engineering
Management and Management Science
Cognitive Science and Human Factors

2.2.3 Criticisms of the SWEBOK Guide and BOK Efforts in General

Although there has been a general consensus concerning the SWEBOK Guide from a large number of practitioners in the software engineering community, there have also been some high-profile detractors of the document. In some cases, those critics assert that the field of software engineering is not yet mature enough for there to be a consensus on a body of knowledge for the field.

The most well-known example of such criticism is a May 2000 ACM task force report assessing SWEBOK and other body of knowledge efforts [32]. This report was authored by David Notkin of the University of Washington and Mary Shaw of Carnegie Mellon University, two highly respected software engineering researchers, and Michael Gorlick of The Aerospace Corporation. This task looked at two major body of knowledge projects that existed at the time: SWEBOK and *The Australian Computer Society Core Body of Knowledge for Information Technology Professionals* (http://www.acs.org.au/national/pospaper/bokpt1.htm). This is from the report's executive summary: "Our study and analysis has led us to the conclusion that the current software engineering body of knowledge efforts, including SWEBOK, are at best unlikely to achieve a goal of critical importance to ACM: the ability to provide appropriate assurances of software quality for software systems of public interest . . . we are uncertain whether, at present, there exists any process to articulate a core body of knowledge in software engineering that will directly contribute to the solution of the software quality problem."

The specific points made by the authors of the report in their assessment of SWEBOK are:

- The Guide is too closely tied to textbooks, which provide an inadequate view of software engineering, as they are by definition targeted to a student audience.
- SWEBOK does not distinguish among potential roles within a software engineering project when discussing the body of knowledge that is required.
- SWEBOK does not address the knowledge for different software application domain areas.
- Since the project sponsors are in "our understanding . . . must make substantial financial contributions to the SWEBOK effort . . . almost certainly disenfranchises some companies and significantly harms the potential authority that SWEBOK might otherwise hold."
- Since the authors feel that the initial SWEBOK development effort is flawed, any process for updating the results would be based on that effort and would therefore be unlikely to succeed.

The report goes on to recommend that the ACM Council, the society's governing board, withdraw ACM from the SWEBOK effort, which they did, in June 2000 (more details are provided in Section 2.5).

Another prominent critic of the SWEBOK Guide is Cem Kaner. Dr. Kaner is a Professor of Computer Science at the Florida Institute of Technology as well as a lawyer, author of a book on software testing, and consultant. In [29], he states some of his concerns about SWEBOK from a legal point-of-view:

"The SWEBOK unconditionally endorses the IEEE Standard 829 for software test documentation. . . We are not aware of scientific research that demonstrates that Standard 829 is a good method, or a better method than others, or desirable under studied circumstances. Standard 829 is in the Body of Knowledge because it won a popularity contest—it was endorsed (or not strongly enough opposed) by the authors of SWEBOK and those relatively few [less than 500 total] people who chose to participate in the SWEBOK drafting and review process so far. . . What is the consequence? A software engineer who recommends against the use of Standard 829 puts herself at [legal] risk. . . An official Body of Knowledge creates an orthodoxy that we do not have today. If the orthodox practices are not well founded in science, as so much of SWEBOK seems not to be, the evolution of the field from weak (but orthodox) practices to better ones will be the subject of lawsuit after lawsuit. And the results of those suits will be determined by non-engineers who don't understand the practices."

Further criticisms of SWEBOK by Dr. Kaner can be found at his web site http://www.kaner.com.

Although it is clear that SWEBOK does not have the general consensus its sponsors have sought, it should also be noted that are also a great number of proponents of the SWEBOK Guide. Also, ACM's current record related to body of knowledge issues for software engineering is not as clear cut as it might seem, as they are working on a body of education knowledge project which has had some relation to SWEBOK (Section 2.3) and are supporting certification exams in software engineering, which by definition require a body of knowledge (Section 3.1). The fact also remains that despite the opposition of the ACM Council and others, software engineers are currently being licensed or undergoing national certification in several countries (or parts thereof), the qualifications for such licensing or comprehensive certification must be assessed against a body of knowledge, and that SWEBOK is the most prominent body of knowledge artifact that currently exists.

2.3 Software Engineering Degree Programs and Accreditation

Accreditation of educational degree programs in a particular country is usually performed either by organizations in conjunction with professional societies, or directly by the societies themselves. For instance, engineering degree programs in the

United States are accredited by the Engineering Accreditation Commission (EAC) of the Accreditation Board for Engineering and Technology, Inc. (ABET).

The first step in licensing of engineers in a particular jurisdiction is usually the earning of a baccalaureate engineering degree [8]. In some cases, the same organization that oversees licensing also accredits degree programs. So, accreditation criteria often influences licensing examinations, or vice versa, especially if (as is the case in the United States) one of those examinations is taken immediately upon graduation.

Undergraduate degree programs in software engineering have been slow to develop in most countries, however, there are now at least 60 currently existing worldwide, including at least 25 in the United States as of September 2003 (up from 21 the year before). The Working Group on Software Engineering Education and Training (WGSEET) (http://www.sei.cmu.edu/collaborating/ed/workgroup-ed.html) has recently begun tracking these programs (and ones in development), and has identified ones in eight different countries. (WGSEET acknowledges that at this point their list is incomplete, and that there are likely many more baccalaureate programs than have currently been identified by them.) Accreditation in some of these countries is described below.

Starting with the University of Sheffield in 1988, the first baccalaureate software engineering programs appeared in the United Kingdom, where the British Computer Society (BCS) and the Institution of Electrical Engineers (IEE) have worked together for over a decade to promote software engineering as a discipline; *A Report on Undergraduate Curricula for Software Engineering*, which was jointly developed and published in 1989 [16], coincided with the appearance of the first undergraduate programs in the UK, but was virtually ignored outside of the United Kingdom. National certification of software engineers through the granting of Chartered Engineer status by BCS started shortly thereafter. As of 2000, there were at least 15 accredited undergraduate software engineering degree programs in the UK. (For details concerning software engineering in the United Kingdom, please consult Thompson and Edwards' excellent article on the subject [37].) Ireland also accredits engineering programs through a similar process, with the first four schools receiving software engineering accreditation in 2001.

Australian universities have created a number of bachelor's degree programs starting in the 1990s. Accreditation of engineering degree programs is granted by IEAust (Institution of Engineers, Australia), which also does certifies professional engineers nationwide. When an engineering program is first created, it needs to obtain provisional accreditation, and then seeks full accreditation once the program has graduates. (This is also the process used by New Zealand, which has accredited software engineering programs as well.)

By 1997, software engineering undergraduate programs also started to appear in Canada, with the first programs receiving accreditation in 2001. By the 2002–03

school year, there were six accredited programs, and two others being reviewed for accreditation. Accreditation in that country is done by Canadian Engineering Accreditation Board (CEAB), which is part of the Canadian Council of Professional Engineers, which is the entity which licenses professional engineers there.

The first undergraduate software engineering program in the United States was started at the Rochester Institute of Technology in the fall of 1996. In the late 1990s, ABET approved criteria for accrediting software engineering under its Engineering Accreditation Commission. The first undergraduate software engineering programs were considered in the 2002–03 accreditation cycle; at least four schools have publicly stated that they were visited by ABET in the fall of 2002.

It is of interest to look at the ABET/EAC software engineering criteria in more detail, since (as will be seen) it does not have the close relationship to licensing or national certification that the other countries mentioned here do. The ABET/EAC criteria [1] contains eight general criteria, of which Criterion 4 (Professional Component) and Criterion 8 (Program Criteria) specific address requirements for specific curriculum content.

Criterion 4 states that "The professional component must include: (a) one year of a combination of college level mathematics and basic sciences (some with experimental experience) appropriate to the discipline; (b) one and one-half years of engineering topics, consisting of engineering sciences and engineering design appropriate to the student's field of study. . ." Note that this means that the continuous mathematical subjects (e.g., calculus and differential equations) taken by most engineering disciplines do no necessarily need to be taken by software engineers. Also, since ABET allows computer science courses to be used as engineering sciences, software engineering majors are not required to take traditional engineering sciences such as statics and thermodynamics. (However, as will be seen later, the licensing examination for graduating engineers in the U.S. still does require continuous mathematics and traditional engineering sciences.)

Criterion 8 specifies criteria for each individual engineering discipline. The curriculum section of the software engineering criteria states that "The curriculum must provide both breadth and depth across the range of engineering and computer science topics implied by the title and objectives of the program. The program must demonstrate that graduates have: the ability to analyze, design, verify, validate, implement, apply, and maintain software systems; the ability to appropriately apply discrete mathematics, probability and statistics, and relevant topics in computer science and supporting disciplines to complex software systems; and the ability to work in one or more significant application domains." Note that the "application domain" section of the criteria addresses one of the concerns expressed by Notkin, Gorlick and Shaw in their report to ACM concerning SWEBOK.

The lead "society" for accrediting software engineering programs within ABET is CSAB (http://www.csab.org), a joint ACM/IEEE-CS organization. ACM and the Computer Society are also collaborating on the development of a curriculum model; the Computing Curricula-Software Engineering (CCSE) project (which was formerly the aforementioned SWEEP) is intended to provide detailed undergraduate software engineering curriculum guidelines which could serve as a model for higher education institutions across the world. The first major component of this project was the development of Software Engineering Education Knowledge (SEEK) [35], a collection of topics considered important in the education of software engineering students. SEEK was created and reviewed by volunteers in the software engineering education community. The SEEK body is a three-level hierarchy, initially divided into knowledge areas (KAs). Those KAs are then further divided into units, and finally, those units are divided into topics.

Each topic in SEEK is also categorized for its importance: Essential, Desired, or Optional. There are currently over 200 essential topics which under a North American educational model. Essential topics are also annotated with indicators from Bloom's Taxonomy in the Cognitive Domain [12] to show the level of mastery expected. SEEK only uses three of the six Bloom Taxonomy values: knowledge, comprehension, and application.

SEEK is important in relative to accreditation in that if it is adopted by the major computing societies, then an argument can be made that an accredited software engineering program should be following its guidelines, in the same way that the SWEBOK Guide might be used in relation to licensing. A worldwide survey of baccalaureate software engineering programs [11] revealed that many of them are using SEEK—even though it is still only in draft form—as an instrument to determine if the proper core software engineering knowledge is being addressed in their respective curricula.

In fact, SEEK and SWEBOK share a number of similarities, including a great deal of overlap in their respective knowledge areas, although they have different requirements and target audiences. These similarities led to the developers of the two projects to hold a workshop to suggest improvements to both artifacts at the 2002 Software Technology and Engineering Practice (STEP) conference in Montreal. The STEP post-conference proceedings contained three papers developed as an outcome of the workshop, including a preliminary mapping of SWEBOK to SEEK [15]. It is also interesting to note that ACM (a co-sponsor of SEEK) is still playing an indirect role in the development of SWEBOK.

There is little controversy today over the concept of software engineering as an academic discipline, although there are still are some individuals that believe that having the such degrees only at the graduate level is essential for providing the proper depth in computer science and other related disciplines. (A panel discussing the rel-

ative merits of undergraduate and graduate software engineering degree programs, chaired by Dewayne Perry of the University of Texas at Austin, was held at the International Conference on Software Engineering in May 2003.) There are also political issues at many higher education institutions involving the distribution of often-scarce resources if a new program such as software engineering is established, just as computer science faced these very same issues 30–40 years. However, there are already a number of accredited software engineering degree programs in several countries, with the United States about to join that group.

A more detailed look at software engineering education and training can be found in the *Encyclopedia of Software Engineering* article on the subject [9].

2.4 Legal Issues in Professional Licensing

Licensing has existed for thousands of years. For instance, a farmer might need a permit from the city to sell wares there. By the 17th century, London required for various craftsmen such as printers and clockmakers to a serve a seven-year apprenticeship before being allowed to ply their trade there. By the 19th century, doctors and lawyers required a *professional* license to practice in most legal jurisdictions. The primary reason behind professional licensing has been the health, safety and welfare of the public. For instance, a doctor should be licensed in order to be entrusted with a patient's medical care, while a lawyer's license attests both knowledge of the law and an adherence to ethical standards to any prospective clients. By the 20th century, the licensing and registration of professional engineers had emerged. Ford and Gibbs list manicurists, amateur boxers and embalmers among over 30 professions which require licenses in the state of California [21].

Frailey notes that in such cases, the risk to the public is high enough to warrant professional licensing [23]. It could easily be argued that many of these same risks are present in software engineering as well; for instance, a software failure can cause a plane to crash, safety protocols to be compromised, or there could be a miscalculation of millions of dollars in a banking transaction due to an incorrect algorithm.

So on face value, it would appear that at least three of the four facts that Frailey has proposed as a requirement to determine the appropriateness of licensing are present: software engineering is emerging as a discipline, it has been long-established as a profession, and that there would be a benefit for licensing software engineers, as long as the fourth criteria—that software engineering is sufficiently established to justify the effort to establish them—is present. Of course, this is also at the heart of the discussion related to whether or not there is a body of knowledge for software engineering (as previously discussed).

If a profession has specific and generally agreed-upon practices, then this provides a basis for determining whether or not someone licensed in that profession has acted

properly in a particular situation. For instance, if a doctor treats a patient, and the patient dies, but it is determined that the physician followed established medical practices in that particular case, then doctor is not legally liable for any actions taken, despite the patient's death. Proponents of the SWEBOK Guide claim that it contains such generally-agreed upon practices for software engineering, while others such as Notkin, Shaw, and Kaner disagree with that assertion.

The liability issue is one that has caused great concern to some segments of the computing community. Kaner writes that

> "For several years, computer malpractice has been a losing lawsuit because to be sued for malpractice (professional negligence), you must be (or claim to be) a member of a profession. Software development and software testing are not professionals as this term is usually used in malpractice law. Therefore, malpractice suits programmers and tester fail. . .
> "So why does it matter whether malpractice is a viable type of lawsuit? Malpractice suits are more serious than suits for breach of contract or simple negligence. . . Licensing [of software engineers] will lead to one thing: malpractice liability. If a state government declares us a profession and starts licensing us, that state's courts will accept us as professionals, and that means they will allow lawsuits for computer malpractice." [28]

If Kaner's view is accurate, then the legal shielding provided by licensing might well be outweighed by the damage to licensed software engineers that would be caused through increased liability through now being open to malpractice suits. An ACM Task Force on safety-critical software chaired by John Knight of the University of Virginia and Nancy Leveson of MIT (with Kaner as one of its other four members) addressed additional concerns related to the malpractice issue, stating:

> "The process of determining that an act constitutes malpractice is only partially driven by engineers. In a typical lawsuit for malpractice, an injured or economically harmed person sues the engineer, often as part of a broader lawsuit. The person will bring evidence that the engineer acted in ways, or made decisions, that were not up to the professional standards of software engineering. This evidence will be evaluated by lawyers, liability insurance company staff and lawyers, jurors, and judges. None of these people are engineers. If juries find that certain conduct is negligent, malpractice insurers will probably advise their insureds (engineers) against engaging in that conduct and may well provide incentives, in the form of lower insurance rates, for engineers who adopt recommended practices or who do not engage in counter-recommended practices. Over time, the determination of what constitute good engineering practices may be driven more by the courts and insurance companies than by engineers." [30]

Still another issue to be considered is what legal obligations an engineering licensing board has as far as regulating the use of the term "engineer". That is, if the law

requires all individuals using the term “engineer” to be licensed, then the issue becomes whether to have a process by which someone can gain the legal right to use the term “software engineer” when advertising for services, or allow no one to do so. This was the issue in Texas at the time licensing of software engineers began there. However, if a particular jurisdiction such as California is only required to license particular specified engineering disciplines, then the choice is whether to allow anyone to use the term “software engineer”, or to regulate its use.

In either case, the question for a legal jurisdiction is: should it specify criteria by which some people can call themselves a software engineer, and some not, and if so, what that criteria will be.

2.5 Pros and Cons of Licensing Software Engineers

There is probably no issue in the computing field as controversial as the licensing of software professionals. Without a doubt, it has caused the biggest public disagreement to date between ACM and IEEE-CS, the world’s largest computing societies, and has left a major scar in the software engineering field that exists to this day. It is a political, philosophical and emotional issue without equal in the computing community.

Several well-known software professionals have written in favor of licensing. David Lorge Parnas of the University of Limerick, one of the pioneers of software engineering, writes:

> In spite of four decades of calls for software development to be viewed as engineering, regulation of software development professionals has been neglected by both the engineering authorities and the computer science community... The public has suffered from this neglect. There are books full of stories about the damage caused by software errors. Delayed and canceled projects are rampant. In most of these cases, the failures can be associated with the failure to apply simple, well-accepted design principles. In many cases, the programmer had never heard of those principles and in other cases, they knew the principles but did not have the discipline to apply them. As long as anyone is free to identify his/herself as a “software engineer,” these problems will remain [33].

This is from Dennis Frailey, senior fellow at the Raytheon Systems Company and adjunct Professor of Computer Science at Southern Methodist University, who has held several different positions in both ACM and IEEE-CS:

> [S]oftware is no longer the cocoon it was when I first entered the field in the 1960s... Now the world notices us, and, more importantly, it cares about what we do and expects us to live by certain standards of professional competence, responsibility, and ethics... Licensing will, however, improve our well-being as a society—if done effectively and properly. It already serves as a catalyst to focus

> attention on establishing software engineering as a discipline and profession... Because licensing forces us to define, codify, and organize our field, it might improve the overall quality of the software engineering discipline ... it could reduce the chances of unqualified, incompetent practitioners building inappropriate software in systems that could harm people [24].

There are a few people in the computing community that oppose licensing of any type of professional. Tom DeMarco, another pioneer in the software engineering field, writes, "I find it morally and ethically repugnant that anyone should prohibit anyone else from seeking to sell his or her services to those who would willingly buy them"; he later goes on to include (U.S.) state bar associations (which regulate lawyers) in that group [19]. However, most people who oppose the licensing of software engineers are not against professional licensing in more established professions such as medicine and law.

As has been previously discussed, there are many legal issues involved in licensing. For instance:

- Some engineering licensing boards are directed to enforce the laws regulating the use of "engineer" as a job description. With the number of software engineers in the United States alone projected to go over one million sometime during this decade, this poses a problem which some licensing boards can no longer ignore.
- Some licensed engineers are faced with have to certify an embedded system where every component of the system has individually been designed by a registered professional engineer—except the software.
- Some jurisdictions in the U.S. require the professors in an accredited engineering program to themselves be licensed professional engineers, which means accreditation can depend on licensing.
- There is an expectation by the graduates of an accredited undergraduate engineering program that there is a path by which they can become licensed; it is therefore difficult to have numerous accredited software engineering degree programs without having a licensing mechanism for the graduates of those programs.
- The licensing of software engineers can lead to increased software malpractice suits, and possibly lead to the courts (as opposed to software professionals) determining what constitutes good software engineering).

There are also economic concerns. Many software professionals believe that licensing would be seen an unnecessary governmental intrusion that would drive companies away from any state which required software engineering registration. However, that does not appear to be the case after several years of licensing in Texas [10].

As previously stated, ACM and IEEE-CS approved the creation of SWECC in late 1998. However, another important event also occurred during that same timeframe: the Texas Board of Professional Engineers began licensing software engineers, the first state to do so [5]. Partially in response to the Texas Board's actions, Barbara Simons, then president of ACM, appointed in March 1999 an advisory panel to make recommendations to the ACM regarding the licensing of software engineers. On 15 May 1999, the ACM passed the following motion based on a report from the panel:

> ACM is opposed to the licensing of software engineers at this time because ACM believes that it is premature and would not be effective at addressing the problems of software quality and reliability [2].

Some ACM Council members then provided their viewpoint in the February 2000 issue of *Communications of the ACM*; however, it is interesting that of the four responses that appeared in the "Forum" section of the May 2000 issue, three took issue with the ACM Council's decision, while the fourth supported it, but anonymously [7].

Subsequent to their decision on licensing, the ACM Council commissioned two other reports: the aforementioned one on the body of knowledge, as well as a report focusing on the potential licensing of software engineering working on safety critical systems (the aforementioned ACM Task Force chaired by Knight and Leveson). The latter report [30] recommends that "No attempt should be made to license software engineers engaged in the development of safety-critical software using the existing [United States] PE mechanism", that body of knowledge efforts should not be pursued, and that instead educational efforts should be increased.

Finally, as a result of these events, the ACM Council passed on 30 June 2000 (the last day of that Council's term) a motion to withdraw from SWECC, because, in the Council's opinion "SWECC has become so closely identified with licensing of software engineers under a professional engineer model" [3]. The July 2000 issue of *Forum for Advancing Software engineering Education* (*FASE*) contains several articles related to the ACM reports and their withdrawal from SWECC, with comments by several noted computing professionals, including Dennis Frailey, who was one of the ACM representatives on SWECC. Frailey opposed ACM's actions in withdrawing from SWECC, stating that:

> The work of over 400 volunteers from about 50 countries has been cast in doubt on the basis of weak and often incorrect rationale… I am also disappointed at the exclusionary process by which this decision was reached. The Council and its task forces chose not to consider the views and insights of ACM's appointed SWEcc representatives and project leads—the people actually doing the work—even after we offered to provide information and to assist in the discussions and

> deliberations. The draft rationale for this decision contains incorrect assumptions and factual errors that we could easily have corrected and that may well have influenced the Council's decision [22].

The September 2000 and September 2001 issues of FASE each had a series of follow-up articles on the withdrawal; the issues are available through the FASE website at http://www.cs.ttu.edu/fase. All of the ACM-related documents can be found at http://www.acm.org/serving/se_policy. *Communications of the ACM* had a section in its November 2002 issue with several articles on the licensing issue.

The joint ACM/IEEE-CS software engineering curriculum effort under SWECC went forward under the new CCSE acronym, but only after a year's delay. As has been discussed, the Computer Society went forward with the SWEBOK Guide as scheduled following ACM's withdrawal from the project.

2.6 Examples of Licensing

This section examines how licensing of engineers has been implemented in Canada and the United States, and how in some cases, it is been to be applied to software engineers.

2.6.1 Canada

In Canada, most engineers must be licensed in order to practice engineering. This licensing of professional engineers is done on the province and territorial level. The Canadian Council of Professional Engineers, according to its website at http://www.ccpe.ca, is "the national organization of the 12 provincial and territorial [bodies] that regulate the practice of engineering in Canada and license the country's more than 160,000 professional engineers."

The first Canadian province to provide licensing guidelines for software engineers was Professional Engineers Ontario (PEO), in 1999. (Previously, software practitioners had been assessed by PEO on an individual basis to see if they qualified for a Professional Engineer (PEng) license.) The CCPE has since adopted guidelines which can be used by each of the twelve licensing bodies within the Council.

The *Guideline on Admission to the Practice of Engineering in Canada* specifies the PEng admissions requirements which apply to all of the seventeen engineering disciplines currently licensed by CCPE. In order to be licensed, applicants must:

- Be academically qualified;
- Have obtained sufficient acceptable engineering work experience in their area of practice;
- Have an understanding of local practices and conditions;

- Be competent in the language of their jurisdiction of practice [Canada is bilingual];
- Be of good character; and,
- Demonstrate an understanding of professional practice and ethics issues. ([17], p. 3)

Academic qualification usually comes from having an engineering degree accredited the Canadian Engineering Accreditation Board, which is a part of CCPE. For those not holding such a degree, a series of examinations is required.

Upon graduation, applicants are expected to enroll in the Engineer-In-Training (EIT) program of the CCPE while fulfilling their work experience requirements, which takes at least four years to complete. The work experience for each applicant should provide some general capabilities in areas such as management and communication skills, and some specific software engineering capabilities from a set of areas (which perform a similar function to the Knowledge Areas of the SWEBOK Guide).

At the end of the EIT period, applicants also are required to pass an examination on ethics and the legal concepts related to professional engineering practice, which should have also been a part of the applicant's work experience.

2.6.2 United States

Unlike Canada, in the U.S. engineers working for a corporation are often covered by an industrial exemption which only requires a license for those engineers who "sign off" (provide a seal of approval and thus incur legal liability) on projects sold to the public. Also, the requirement such legal liability is usually limited to those projects which are for a specific customer. That is, if a software product is mass-marketed to the public, then a professional engineer is not required to sign off on the product. However, independent engineering contractors (i.e., consultants) must be licensed.

This means that the percentage of engineers licensed in the United States is relatively small, and varies from discipline to discipline. Ford and Gibbs [21] state that the percentage licensed range from less than ten percent for electrical and chemical engineers to over 40% for civil engineers. Most expert predict that the percentage of licensed software engineers will be even less—perhaps 5%. Still, with over a million software engineers expected in the U.S. by 2010, 5% is still a significant number of people.

Since professional licensing is not delegated to federal government by the United States Constitution, the each state licenses professional engineers. The scope and requirements of the licensing boards for the various states can widely vary; however, the basic format in contained in the *Model Law* document [31] of the National

Council of Examiners for Engineering and Surveying (NCEES), which creates engineering licensing exams for most of the United States. (The NCEES board consists of representatives of the various state licensing boards.) The general path to licensure as a professional engineer (PE) is as follows:

- Obtain a degree from an ABET/EAC accredited program,
- Pass the NCEES Fundamentals of Engineering (FE) examination given to people who are about to graduate or have recently graduated with a bachelor's degree,
- Work a minimum of four years under the supervision of a Professional Engineer,
- Pass the NCEES Principles and Practices of Engineering (P&P) exam most related to the engineering work being done by the applicant,
- Obtain professional references, and
- Pass an examination of the code of ethics and professional conduct for that particular state.

The FE exam is divided into morning and afternoon sessions. The morning part of the FE exam is the same for all applicants and includes mathematics (calculus, differential equations, linear algebra), lab sciences (chemistry and physics), and engineering sciences (e.g., statics, materials, and thermodynamics). Note that although there is an implication is that a student graduating from an accredited engineering program in the United States should have the educational background sufficient for passing the FE morning section, although the ABET criteria and the morning section content may be significantly different—and in the case of software engineering, they are. It is not surprising then that unlike the other countries discussed here, there are two completely independent entities doing the accreditation and the licensing.

The FE afternoon session can be discipline-specific (for a limited number of disciplines), or the applicant can take a general examination which in many ways is an extension of the morning session.

NCEES will not offer P&P or FE afternoon examinations for engineering disciplines that do not have at least one accredited program in that area, thus there can be no action until ABET accredits the first software engineering programs (expected in July 2003). At that time, at least ten state licensing boards would have to request that NCEES develop an exam in order for such a project to be considered. However, for at least the time being, all aspects of the NCEES licensing exams are somewhat divergent from software engineering curriculum content.

Texas is the only state that currently licenses software engineers as PEs. It has been doing so since 1998. Because there are no software engineering licensing exams, the Texas Board of Professional Engineers has been using its waiver clause (which is available for all engineering disciplines licensed by the board) to license

software engineers. In this case, practitioners with 12 years of experience and an ABET-accredited degree, or 16 years of experience for those with a related degree, are eligible to be licensed. All applicants in Texas, regardless of waiver status, must pass an engineering ethics examination [4]. As of 10 July 2003, there were only 48 licensed PEs in software engineering in Texas.

The Illinois legislature has passed legislation requiring that there a P&P examination in software engineering available to applicants in that state by January 1, 2005 [26]. Since NCEES will undoubtedly not have such an exam available by that time, it is unclear on what will happen. (Some states do provide their own P&P exams in certain cases to take in the place of their NCEES equivalents, so this is a possibility.)

2.7 Examples of National Certification

2.7.1 United Kingdom

Note: The information for the United Kingdom is primarily from [37], which in turn used material from the British Computer Society website http://www.bcs.org.uk.

Technically, the UK certifies engineers through the professional bodies; this is also the case with other professions such as law and medicine. However, since each of these bodies has a Royal Charter as a mechanism which is used as a means of ratifying the professions, and a profession is granted chartered status by the Government on satisfying particular criteria specified in order to protect the public from unqualified or illegal practices, the professional societies are doing something very similar to the government-sanctioned licensing of professionals done in the U.S. and Canada. For people in the computing field (including software engineering), this is the British Computer Society.

The BCS is licensed by the UK's Engineering Council to nominate any of its members that are academically qualified and appropriately experiences to be Chartered Engineers (CEng). In the UK, anyone can call themselves an engineer, but can only use the CEng designation if it granted. BCS actually grants CEng status in Information Systems, which is a broader discipline than software engineering.

To gain admission to the BCS (the first step towards becoming a Chartered Information Systems Engineering), the society has a set of examinations and assessments; however, the BCS gives partial or full exemption from its examinations for additional academic credentials.

Members registered as Chartered Engineers may also apply for the qualification "European Engineer" through Federation Europeene d'Associations Nationales d'Ingenieurs (FEANI), which has representatives from 22 European countries.

2.7.2 Australia

The certification of professional engineers in Australia is done through IEAust http://www.ieaust.org.au, which (as with the British Computer Society) also accredits degree programs. IEAust also has several levels of membership, of which Chartered Professional Engineer (CPEng) is both the highest possible level and the one which is analogous to the professional engineer designation in other countries. However, IEAust does not have the same relationship with its government as BCS does with the British government, since the Australian government supports self-regulation of professionals through the professional societies.

As with the BCS, an applicant for a CPEng must first be a member of IEAust at another grade level [27]. For instance, a Member IEAust (MIEAust) must have a degree from an accredited engineering program, have a minimum of three years of professional experience, and have the support of a member at an equivalent of higher grade. An applicant for a CEng must also submit background information in the form of an Engineering Practice Report (EPR), which must be verified as satisfactory by IEAust. Once this is done, the applicant will be invited to a one-hour professional interview conducted by CPEng members from the applicant's chosen engineering discipline. This interview will be a peer review of the competencies that the applicant has claimed in the EPR, plus test knowledge of the Institution's Code of Ethics. Those individuals receiving the CPEng designation can request to be placed on the National Professional Engineers Register (NPER).

Currently, software engineers are not allowed to receive the CPEng designation or be placed on the NPER. In 2001, the Australian Computer Society (ACS) and IEAust formed a Joint Software Engineering Board, and developed a discussion paper on the topic of software engineering as a professional discipline. ACS is currently petitioning IEAust to allow software engineering to be added to the list of approved engineering disciplines; [27] lists "Information, Telecommunications and Electronics Engineering" (whose description includes software engineering) as a general area of engineering practice, but also notes that registration on the NPER under this engineering is not available as of yet.

2.7.3 Ireland

The Institution of Engineers of Ireland certifies its Chartered Engineers in a manner vary similar to that of IEAust. However, unlike Australia, IEI has already started chartering software engineers during the last few years. Unlike their counterparts in Australia and the UK, IEI does classify software engineers separately.

3. The Certification of Software Developers

3.1 Institute-Based Certification Programs

The three examples of "institute-based" certifications here only have one example that is actually called an institute. However, they all are part of that class of certification outside of either national or company-based certification, programs that are either done by a private institute created solely for certification, or a professional society sponsoring a certification program which is not targeted toward a particular nation.

3.1.1 ASQ

Founded in 1946, The American Society for Quality (ASQ) provides training and certification in a variety of quality areas, including one specifically for software professionals, the Certified Software Quality Engineer (CSQE). This program, according to a page on the ASQ website http://www.asq.org/cert/types/index.html, is "Designed for those who have a comprehensive understanding of software quality development and implementation; have a thorough understanding of software inspection, testing, verification, and validation; and can implement software development and maintenance processes and methods."

In order to take the CSQE exam, an applicant must have at least eight years of experience, some of which can be waived due to the person's educational background.

The CSQE is one of two examples shown here (the other being the Microsoft Certified Systems Engineer) where the term "engineer" is granted on a certification, but its use in some jurisdictions (such as certain states or provinces in North America) might be illegal. If intending to practice in such jurisdictions, Kaner suggests contacting ASQ to see if an alternate name to the certification can be provided [28].

3.1.2 CSDP

The IEEE Computer Society has developed a competency recognition program for software professionals [38]. During the development, the program was known as the "Certified Software Engineering Professional Program", but due to the same legal issues mentioned above regarding the use of the term "engineer", the name was ultimately changed to Certified Software Development Professional (CSDP). Despite the name change, it is still software engineering knowledge being tested (although SWEBOK was *not* used for the test specifications).

The overall certification program includes requirements on education, professional experience, passing an examination, and continuing education. To be eligible to take the exam, a candidate must have, at minimum, a bachelor's degree and 9000 hours of

software engineering experience. (Since 9000 hours translates to a person working 45 hours per week, 50 weeks a year for four years, the CSDP can be thought of as roughly analogous to a P&P exam given in the United States by NCEES.) After passing the exam, certificate holders will be required to obtain a number of approved continuing education hours over a three year period to renew their certification.

The initial CSDP examinations were developed from 1999 to 2001, and beta tested in July 2001. There have since been two years of regular test cycles, and there are now several hundred CSDP certificate holders. Information concerning training and test specifications can be found at CSDP website http://www.computer.org/certification. There is also a well-trafficked (356 members, as of July 2003) Yahoo group (ieee_csdp) started by the IEEE-CS Seattle chapter. Despite some initial disappointment that the term "engineer" is not used in the certification, the CSDP seems to be gradually gaining momentum.

The IEEE-CS Professional Practices Committee, which created and oversees the CSDP exam, is currently considering the eventual creation of an entire suite of related certification exams for software professionals, including an exam for new graduates (roughly analogous to the FE exam). These specialization exams would be targeted towards various job functions (e.g., software project manager), and application domain specialty (such as for safety-critical systems).

3.1.3 ICCP

The Institute for Certification of Computing Professionals (ICCP) is located in Des Plaines, Illinois, USA, and is under the direction of seven constituent societies, including ACM. ICCP has certified over 55,000 people as Certified Computer Professionals over their 30 years of existence. ICCP offers an exam in a number of software-related areas, including one software engineering which includes the following topics:

- Computer System Engineering
- Software Project Planning
- Software Requirements
- Software Design
- Programming Languages and Coding
- Software Quality Assurance
- Software Testing Techniques
- Software Maintenance and Configuration Management

It is interesting to note the similarities between the Knowledge areas of the SWEBOK Guide and the above topics, since ACM has publicly stated it concerns related

its concerns about defining a body of knowledge sufficient for licensing examinations, and yet has continued to offer a certification exam on similar material.

As with the ASQ Certified Software Quality Engineer exam, an applicant must have work experience, the amount of which can be reduced according to the person's educational background.

3.2 Company-Based Certification Programs

There are a number of companies dealing in mass-marketed computer systems that offer certification programs, including Microsoft, Oracle, Cicso, Novell and Apple. Most of these software-related certifications actually focus on network or systems administration rather than on software development. In recent years, the Microsoft Certified Solution Developer (MCSD) program has emerged as the company-based certification that has most focused on software life-cycle issues. When migrating the MCSD program to .NET, Microsoft went even further, separating it into two programs: MCSD, which focuses on the analysis and design of software for web applications, windows applications and XML web services and servers in ether Visual Basic or Visual C#, and Microsoft Certified Application Developer (MCAD), which focuses on coding, testing and maintenance of these same type of applications.

The MCSD requires the passage of four core exams, one each for web applications, windows applications and XML web services and servers in either Visual Basic or Visual C#, and a fourth in Solution Architectures, plus one of three elective exams. No practical experience is required, although two years of experience software design and analysis is recommended.

As of July 2003, there are almost 45,000 MCSD holders, of which about 1700 were certified after the program migrated to .NET. These are among over 1.5 million who have earned the title Microsoft Certified Professional (MCP). One of the exams in this suite, the Microsoft Certified Systems Engineer (MCSE), is another example of a certification whose use of the term "engineer" may cause a problem in some jurisdictions. (Many other companies also use engineer in some of their certifications.) In Canada, the problem was solved when CCPE and Microsoft came to an agreement where Microsoft would advise their Canadian certificate holders to use the MCSE acronym rather than the full name of the certification [18].

3.3 Conclusions and Future Directions

The licensing and certification of software engineers by the professional engineering community, either through engineering professional societies or engineering licensing boards, has become more frequent over the last decade. The United Kingdom, Canada, and Ireland are examples of countries which either perform licensing

through government boards or certification of software engineers through national professional societies. In the United States, only Texas is currently licensing software engineers, with Illinois apparently set to follow suit in 2005.

The formal education for licensing and national certification is often through accredited software engineering undergraduate degree programs. The UK, Canada, Australia, New Zealand and Ireland are all examples of countries which have accredited software engineering programs, with the United States very likely to join them in July 2003. The accreditation and licensing bodies in most of these countries are strongly connected; however, this is not true in the United States, which has caused some significant differences between the content of the accredited degree programs, and that of the Fundamentals of Engineering examination given to new graduates of such programs as the first step towards licensing.

The licensing of software engineers has been very controversial, especially in the United States. The Association for Computing Machinery has come out strongly against licensing, citing liability issues, plus the inability to define a body of knowledge; however, at the same time, ACM has supported certification, including an ICCP software engineering examination implicit covering a general software engineering body of knowledge.

Despite ACM's position, licensing and national certification activities have continued to progress. Because of these ongoing activities the author's personal opinion has always been that the active involvement of software professionals in the licensing and national certification processes benefits the software engineering community, and that positive aspects of working with the licensing entities have outweighed the negative ones [10]. For instance, in Texas, it is likely that without the involvement of a Software Engineering Advisory Committee, the state licensing board would still have started licensing individuals in software engineering, but might well have instead treated the discipline as a subarea of electrical engineering.

So, it is the author's personal hope that ACM will somehow reverse their previous stance and become involved in the licensing and national certification efforts in various countries. At the same time, IEEE-CS and others need to acknowledge that the acceptance of the SWEBOK Guide as a generally-accepted body of knowledge document will be extremely difficult (although not impossible) without the support of the ACM Council.

Besides the continuing licensing and national certification efforts, company-based certification is likely to continue its popularity, especially among those software professionals who focus on programming issues. Other certifications (such as CSDP and Certification Software Quality Engineer) will continue to exist, although the number of such certifications may in some cases be tied to the progress of licensing efforts.

A hierarchical licensing and certification model for software engineers has been suggested by several people over the last few years. Frailey [23] compares his ver-

sion of it to the medical model, where doctors must be licensed in order to practice medicine, and can obtain board certification in specialization areas such as internal medicine or dermatology. For software engineering, such specialization might include specific job functions (e.g., software project manager) or specific application domains such as real-time systems. (The application domain specialization area is one that is not currently addressed by either the knowledge areas of the SWEBOK Guide or in current licensing and certification efforts.) Frailey also notes that medical technician certifications such as for the use of X-Ray machines are roughly equivalent most of the company-based certifications.

This model for licensing is also interesting because it could include most of the licensing and certification efforts that currently exist. For example, the Software Quality Engineer certification could be a specialization area obtained by a licensed software engineer. It may be that through the use of such models, which are different from that used in most engineering disciplines, the unique field of software engineering can find a method of licensing and certification that can be to the most benefit to both the software professional community and the public-at-large.

ACKNOWLEDGEMENTS

The author appreciates the invaluable research help provided by the following individuals: Jocelyn Armarego, David Budgen, Robert Cochran, Ana Moreno, Fred Otto, Michael Ryan, J. Barrie Thompson and Alan Underwood.

REFERENCES

[1] ABET Engineering Accreditation Commission, “Criteria for accrediting engineering programs”, http://www.abet.org/images/Criteria/2002-03EACCriteria.pdf, 3 November 2001.

[2] Allen F., Boehm B., Brooks F., Browne J., Farber D., Graham S., Graham G.J., Hawthorn P., Kennedy K., Leveson N., Nagel D., Neumann P., Parnas D., Wulf B., “ACM panel on professional licensing in software engineering: Report to council”, http://www.acm.org/serving/se_policy/report.html, 15 May 1999.

[3] Association for Computing Machinery, “A summary of the ACM position on software engineering as a licensed engineering profession, final version”, http://www.acm.org/serving/se_policy/selep_main.html, 17 July 2000.

[4] Bagert D.J., “Texas poised to license professional engineers in software engineering”, *ACM Software Engineering Notes* **23** (3) (May 1998) 8–10.

[5] Bagert D.J., “Texas board votes to license software engineers”, *ACM Software Engineering Notes* **23** (5) (September 1998) 7.

[6] Bagert D.J., "SIGCSE survey, TCEA discussion on company-based certification", *Forum for Advancing Software engineering Education (FASE)* **10** (3) (March 2000) (electronic newsletter), http://www.cs.ttu.edu/fase/v10n03.txt.
[7] Bagert D.J., "Communications of the ACM Forum on Licensing Issue", *Forum for Advancing Software engineering Education (FASE)* **10** (5) (May 2000) (electronic newsletter), http://www.cs.ttu.edu/fase/v10n05.txt.
[8] Bagert D.J., Mead N.R., "Software engineering as a professional discipline", *Computer Science Education* **11** (1) (March 2001) 73–87.
[9] Bagert D.J., "Education and training in software engineering", in: *Encyclopedia of Software Engineering*, second ed., John Wiley and Sons, 2002, pp. 452–465.
[10] Bagert D.J., "Texas licensing of software engineers: All's quiet—for now", *Communications of the ACM* **45** (11) (November 2002) 92–94.
[11] Bagert D.J., Ardis M.A., "Software engineering baccalaureate programs in the United States: An overview", in: *Proceedings of Frontiers in Education Conference, Boulder, Colorado, USA*, 5–8 November 2003, submitted for publication.
[12] Bloom B.J., et al. (Eds.), *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain*, first ed., David McKay Co., New York, NY, 1956.
[13] Bourque P., Dupuis R., Abran A., Moore J.W., Tripp L., "The guide to the software engineering body of knowledge", *IEEE Software* **16** (6) (November/December 1999) 35–44.
[14] Bourque P., Dupuis R. (Eds.), *SWEBOK: A Guide to the Software Engineering Body of Knowledge (Trial Version 1.00)*, IEEE Computer Society, Los Alamitos, CA, USA, May 2001.
[15] Bourque P., Robert F., Lavoie J.-M., Lee A., Trudel S., Lethbridge T.C., "Guide to the software engineering body of knowledge (SWEBOK) and the software engineering education knowledge (SEEK)—A preliminary mapping", in: *Proceedings of the 10th International Workshop on Software Technology and Engineering Practice, Montreal, Canada*, 6–8 October 2002, pp. 8–23.
[16] British Computer Society and The Institution of Electrical Engineers, *A Report on Undergraduate Curricula for Software Engineering*, Institution of Electrical Engineers, 1989.
[17] Canadian Council of Professional Engineers, *Guideline on Admission to the Practice of Engineering in Canada*, 2001.
[18] Canadian Council of Professional Engineers, "Canadian Council of Professional Engineers and Microsoft Corp. agree on use of 'Engineer' title", news release, 11 May 2001; Reprinted in *Forum for Advancing Software engineering Education (FASE)* **11** (6) (June 2001) (electronic newsletter), http://www.cs.ttu.edu/fase/v11n06.txt.
[19] DeMarco T., "Certification or Decertification?", *Communications of the ACM* **42** (7) (July 1999) 10 (letter to the editor).
[20] Douglas P., Cocchi T., "Report on analyses of pilot software engineer survey data", Joint Steering Committee of IEEE Computer Society/ACM for Establishment of Software Engineering as a Profession, http://www.computer.org/tab/seprof/survey.htm, 27 March 1997.

[21] Ford G., Gibbs N., *A Mature Profession of Software Engineering, Technical Report CMU/SEI-96-TR-004*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1996.

[22] Frailey D.J., "Statement regarding ACM's withdrawal from SWEcc", *Forum for Advancing Software engineering Education (FASE)* **10** (7) (July 2000) (electronic newsletter), http://www.cs.ttu.edu/fase/v10n07.txt.

[23] Frailey D.J., "Licensing and certification of software engineering personnel", in: *Encyclopedia of Software Engineering*, second ed., John Wiley and Sons, 2002, pp. 452–465.

[24] Frailey D.J., "Software engineering grows up", *IEEE Software* **16** (6) (November/December 1999), pp. 66, 68.

[25] Gotterbarn D., Miller K., Rogerson S., "Software engineering Code of Ethics is approved", *Communications of the ACM* **42** (10) (October 1999) 102–107.

[26] "Illinois Compiled Statutes, Professions and Occupations, Professional Engineering Practice Act of 1989 (amended), amendment effective 1 January 2002, Chapter 225, Statue 325, Section 9", http://www.legis.state.il.us/legislation/ilcs/ch225/ch225act325.htm.

[27] Institution of Engineers, Australia, *Chartered Professional Engineers*, May 2002.

[28] Kaner C., "Computer Malpractice", http://www.kaner.com/malprac.htm, 2000. This is an updated version of a paper with the same title that appeared in *Software QA* **3** (4) (1996) 23.

[29] Kaner C., "Software engineering as a profession after the withdrawal: One year later", *Forum for Advancing Software engineering Education (FASE)* **11** (9) (September 2001) (electronic newsletter), http://www.cs.ttu.edu/fase/v11n09.txt.

[30] Knight J., Leveson N., DeWalt M., Elliot L., Kaner C., Nissenbaum H., *On Licensing of Software Engineers Working on Safety-Critical Software*, Association for Computing Machinery, August 2001, http://www.acm.org/serving/se_policy/safety_critical.pdf.

[31] National Council of Examiners for Engineering and Surveying, *Model Law*, Revised August 2002.

[32] Notkin D., Gorlick M., Shaw M., *An Assessment of Software Engineering Body of Knowledge Efforts*, Association for Computing Machinery, New York, May 2000, http://www.acm.org/serving/se_policy/bok_assessment.pdf.

[33] Parnas D.L., "Licensing of software engineers in Canada", *Communications of the ACM* **45** (11) (November 2002) 94–96.

[34] Pressman R.S., *Software Engineering: A Practitioner's Approach*, fifth ed., McGraw-Hill, Boston, MA, 2001.

[35] Sobel A.E.K. (Ed.), *Second Draft of the Software Engineering Education Knowledge*, 6 December 2002, http://sites.computer.org/ccse/know/SecondDraft.pdf.

[36] Sommerville I., *Software Engineering*, sixth ed., Addison–Wesley, Wokingham, England, 2000.

[37] Thompson J.B., Edwards H.M., "Software engineering in the UK 2001", *Forum for Advancing Software engineering Education (FASE)* **11** (11) (November 2001) (electronic newsletter), http://www.cs.ttu.edu/fase/v11n11.txt.

[38] Tockey S., “IEEE Computer Society develops competency recognition program”, *Forum for Advancing Software engineering Education (FASE)* **11** (9) (September 2001) (electronic newsletter), http://www.cs.ttu.edu/fase/v11n09.txt.
[39] U.S. Department of Labor, “The 2000–10 job outlook in brief”, *Occupational Outlook Quarterly* **46** (1) (Spring 2002) 9–43.

Cognitive Hacking

GEORGE CYBENKO

Thayer School of Engineering
Dartmouth College
8000 Cummings Hall Hanover, NH 03755-8000
USA
george.cybenko@dartmouth.edu

ANNARITA GIANI

Institute for Security Technology Studies
Thayer School of Engineering
Dartmouth College
8000 Cummings Hall Hanover, NH 03755-8000
USA
annarita.giani@dartmouth.edu

PAUL THOMPSON

Institute for Security Technology Studies
Thayer School of Engineering
Dartmouth College
8000 Cummings Hall Hanover, NH 03755-8000
USA
paul.thompson@dartmouth.edu

Abstract

In this chapter, we define and propose countermeasures for a category of computer security exploits which we call "cognitive hacking." Cognitive hacking refers to a computer or information system attack that relies on changing human users' perceptions and corresponding behaviors in order to be successful. This is in contrast to denial of service (DOS) and other kinds of well-known attacks that operate solely within the computer and network infrastructure. Examples are given of several cognitive hacking techniques, and a taxonomy for these types of

ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60002-1

attacks is developed. Legal, economic, and digital government implications are discussed.

1. Introduction

Cognitive hacking refers to a computer or information system attack that relies on changing human users' perceptions and corresponding behaviors in order to be successful [24]. This is in contrast to denial of service (DOS) and other kinds of well-known attacks that operate solely within the computer and network infrastructure. With cognitive attacks neither hardware nor software is necessarily corrupted. There may be no unauthorized access to the computer system or data. Rather the computer system is used to influence people's perceptions and behavior through misinformation. The traditional definition of security is protection of the computer system from three kinds of threats: unauthorized disclosure of information, unauthorized modification of information, and unauthorized withholding of information (denial of service). Cognitive attacks, which represent serious breaches of security with significant economic implications, are not well covered by this definition.

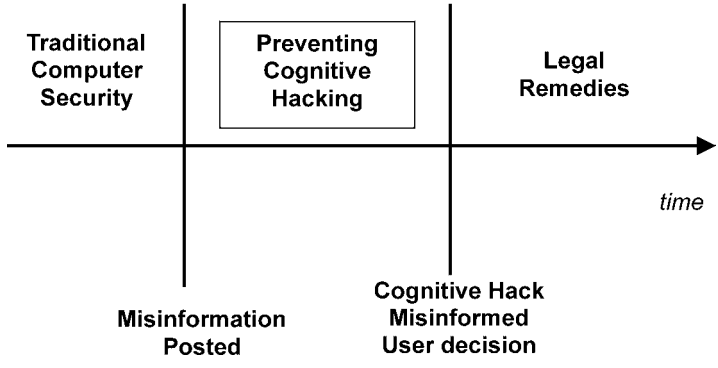


FIG. 1.

In face-to-face interaction with other people, there is normally some context in which to evaluate information being conveyed. We associate certain reliability to information depending on who the speaker is and on what we know of the person. This type of evaluation cannot be transferred to the Web [99]. Anyone can post anything on a Web page with very few limitations. The issue is how to deal with false information on the Web and how to decide whether a source is reliable. People use Web technology for personal and economic reasons, e.g., buying shares or finding a job. What process takes place if a user makes a decision based on information found on the Web that turns out to be misinformation?

Consider the graph below. Most analyses of computer security focus on the time before misinformation is posted, i.e., on preventing unauthorized use of the system. A cognitive hack takes place when a user's behavior is influenced by misinformation. At that point the focus is on detecting that a cognitive hack has occurred and on possible legal action. Our concern is with developing tools to prevent cognitive hacking, that is, tools that can recognize and respond to misinformation before a user acts based on the misinformation, see Fig. 1.

1.1 Background

Computer and network security presents great challenges to our evolving information society and economy. The variety and complexity of cybersecurity attacks that have been developed parallel the variety and complexity of the information technologies that have been deployed; with no end in sight for either. In this chapter, we distinguish three classes of information systems attacks: *physical, syntactic*, and *cognitive*. Physical and syntactic attacks can be considered together as autonomous attacks.

Autonomous attacks operate totally within the fabric of the computing and networking infrastructures. For example, the well-know Unicode attack against older, unpatched versions of Microsoft's Internet Information Server (IIS) can lead to root/administrator access. Once such access is obtained, any number of undesired activities by the attacker is possible. For example, files containing private information such as credit card numbers can be downloaded and used by an attacker. Such an attack does not require any intervention by users of the attacked system; hence, we call it an "autonomous" attack.

By contrast, a *cognitive* attack requires some change in users' behavior, effected by manipulating their perceptions of reality. The attack's desired outcome cannot be achieved unless human users change their behaviors in some way. Users' modified actions are a critical link in a cognitive attack's sequencing. To illustrate what we mean by a cognitive attack, consider the following news report [62]:

> "Friday morning, just as the trading day began, a shocking company press release from **Emulex** (Nasdaq: EMLX) hit the media waves. The release claimed that Emulex was suffering the corporate version of a nuclear holocaust. It stated that the most recent quarter's earnings would be revised from a $0.25 per share gain to a $0.15 loss in order to comply with Generally Accepted Accounting Principles (GAAP), and that net earnings from 1998 and 1999 would also be revised. It also said Emulex's CEO, Paul Folino, had resigned and that the company was under investigation by the Securities and Exchange Commission.
>
> Trouble is, none of it was true.
>
> The real trouble was that Emulex shares plummeted from their Thursday close of $113 per share to $43—a rapid 61% haircut that took more than $2.5 billion off of the company's hide—before the shares were halted an hour later. The damage had been done: More than 3 million shares had traded hands at the artificially low rates. Emulex vociferously refuted the authenticity of the press release, and by the end of the day the company's shares closed within a few percentage points of where they had opened."

Mark Jakob, 23 years old, fraudulently posted the bogus release on Internet Wire, a Los Angeles press-release distribution firm. The release was picked up by several business news services and widely redistributed scale without independent verification. The speed, scale and subtlety with which networked information propagates have created a new challenge for society, outside the domain of classical computer security which has traditionally been concerned with ensuring that all use of a computer and network system is authorized.

The use of information to affect the behavior of humans is not new. Language, or more generally communication, is used by one person to influence another. Propaganda has long been used by governments, or by other groups, particularly in time of war, to influence populations [19,30,34]. Although the message conveyed by pro-

paganda, or other communication intended to influence, may be believed to be true by the propagator, it usually is presented in a distorted manner, so as to have maximum persuasive power, and, often, is deliberately misleading, or untrue. Propaganda is a form of perception management. Other types of perception management include psychological operations in warfare [47], consumer fraud, and advertising [19,77]. As described in Section 1.5, deception detection has long been a significant area of research in the disciplines of psychology and communications.

1.2 Perception Management

As noted by many authors, e.g., [19,28,34,76], perception management is pervasive in contemporary society. Its manifestation on the Internet is one aspect of the broader phenomenon. Not all perception management is negative, e.g., education can be considered a form of perception management; nor is all use of perception management on the Internet cognitive hacking (see definition in the next section). Clearly the line between commercial uses of the Internet such as advertising, which would not be considered cognitive hacking, and manipulation of stock prices by the posting of misinformation in news groups, which would be so considered, is a difficult one to distinguish.

1.3 Computer Security Taxonomies

In 1981 Landwehr provided a discussion of computer system security which has framed subsequent discussion of computer security [55]. His model arose from a consideration of the requirements of military security as automated systems replaced paper-based systems. He postulated that:

> Information contained in an automated system must be protected from three kinds of threats: (1) the *unauthorized disclosure* of information, (2) the *unauthorized modification* of information, and (3) the *unauthorized withholding* of information (usually called *denial of service*).

He concludes his discussion by stating that “Without a precise definition of what security means and how a computer can behave, it is meaningless to ask whether a particular computer system is secure” [56]. If certain uses and sets of data can be completely isolated, then security modeling is tractable, but often this isolation is not possible, in part because the use of resources by a computer system conveys information. Although the analysis Landwehr presented was based on an analysis of secure military computing, he advises that the designer of an application must state in advance the security properties desired of the system, recognizing that systems with different goals and operating in different environments, will have different security needs.

Over the last 20 years earlier attempts in computer security research to develop formal security models designed into the computer system were abandoned as various models were shown to be undecidable [29]. As more and more elaborate security approaches were developed, they were rendered out of date by rapid changes in the computing environment, in particular, the development of the World Wide Web. In recent years dramatic and costly computer viruses, denial of service attacks, and concerns with the security of e-commerce have drawn the attention of researchers in computer security. While these security breaches are a serious concern, cognitive hacking is a form of attack that deserves to receive more attention.

Cognitive hacking is defined here as gaining access to, or breaking into, a computer information system for the purpose of modifying certain behaviors of a human user in a way that violates the integrity of the overall user-information system. The integrity of such a system would, for example, include correctness or validity of the information the user gets from such a system. In this context, the integrity of a computer system can be defined more broadly than the definition implicit in Landwehr's definition of computer security. Smith [87] refers to breaches in computer security as violations of the semantics of the computer system, i.e., the intended operation of the system. Wing argues a similar view [95]. In this sense the World Wide Web itself can be seen as a computer system used for communication, e-commerce, and so on. As such, activities conducted over the Web that violate the norms of communication or commerce, for example, fraud and propaganda, are considered to be instances of cognitive hacking, even if they do not involve illegitimate access to, or breaking into, a computer. For example, a person might maintain a website that presents misinformation with the intent of influencing viewers of the information to engage in fraudulent commercial transactions with the owner of the website.

Some examples of cognitive hacking, such as the manipulation of a Yahoo news item [68], are instances of Landwehr's second threat, the *unauthorized modification* of information [55], but there are many other examples that do not fit Landwehr's taxonomy. This is not surprising, because, as suggested by Landwehr, new applications, i.e., in this case Web services, will have new security needs, which must be understood.

Two broad classes of cognitive hacking can be distinguished: overt and covert. With overt cognitive hacking no attempt is made to conceal the fact that a cognitive hack has occurred. For example, website defacement is a type of overt cognitive hacking. While a website defacer may hope that the defacement is not noticed for as long as possible by a Web page administrator, the Web defacer's modification to the website is a blatant modification that the intended audience will realize immediately is not the unmodified website.

It has been estimated that 90% of attacks on Web pages are total page hacks, where the attacker replaces the entire page of the attacked site [45]. This overt cog-

nitive hacking, while much more prevalent than other covert forms discussed in this chapter, is more of a nuisance to website administrators and an embarrassment to website owners. Covert cognitive hacking, by contrast, is likely to have more significant consequences, because it can influence a user's perceptions and behavior.

Misinformation is often a covert form of cognitive hacking. Misinformation is an intentional distribution or insertion of false or misleading information intended to influence reader's decisions and/or activities. The open nature of the Internet makes it an ideal arena for the dissemination of misinformation [17].

The distinction between overt and covert cognitive hacking is not necessarily unambiguous. For example, the recent hacking of the Yahoo news story [68] was clearly not as blatantly overt as a typical Web defacement. On the other hand, the changes introduced were obvious enough that the suspicions of a careful reader would soon be aroused. It is possible, though, to imagine subtler modifications that would have gone undetected by any reader not already familiar with a more reliable account of the news item.

1.4 Semantic Attacks and Information Warfare

A definition of semantic attacks closely related to our discussion of cognitive hacking has been described by Schneier [85], who attributes the earliest conceptualization of computer system attacks as physical, syntactic, and semantic to Martin Libicki, who describes semantic attacks in terms of misinformation being inserted into interactions among intelligent agents on the Internet [60]. Schneier, by contrast, characterizes semantic attacks as "... attacks that target the way we, as humans, assign meaning to content." He goes on to note, "Semantic attacks directly target the human/computer interface, the most insecure interface on the Internet" [85].

Denning's discussion of information warfare [28] overlaps our concept of cognitive hacking. Denning describes information warfare as a struggle over an information resource by an offensive and a defensive player. The resource has an exchange and an operational value. The value of the resource to each player can differ depending on factors related to each player's circumstances. The outcomes of offensive information warfare are: increased availability of the resource to the offense, decreased availability to the defense, and decreased integrity of the resource. Applied to the Emulex example, described below, Jakob is the offensive player and Internet Wire and the other newswire services are the defensive players. The outcome is decreased integrity of the newswires' content. From the perspective of cognitive hacking, while the above analysis would still hold, the main victims of the cognitive hacking would be the investors who were misled. In addition to the decreased integrity of the information, an additional outcome would be the money the investors lost.

1.5 Deception Detection

Detection of deception in interpersonal communication has long been a topic of study in the fields of psychology and communications [48,12,21,13]. The majority of interpersonal communications are found to have involved some level of deception. Psychology and communications researchers have identified many cues that are characteristic of deceptive interpersonal communication. Most of the this research has focused on the rich communication medium of face-to-face communication, but more recently other forms of communication have been studied such as telephone communication and computer-mediated communication [99]. A large study is underway [14,38] to train people to detect deception in communication. Some of this training is computer-based. Most recently a study has begun to determine whether psychological cues indicative of deception can be automatically detected in computer-mediated communication, e.g., e-mail, so that an automated deception detection tool might be built [98,99].

1.6 Cognitive Hacking and Intelligence and Security Informatics

Intelligence and security informatics [16] will be supported by data mining, visualization, and link analysis technology, but intelligence and security analysts should also be provided with an analysis environment supporting mixed-initiative interaction with both raw and aggregated data sets [89]. Since analysts will need to defend against semantic attacks, this environment should include a toolkit of cognitive hacking countermeasures. For example, if faced with a potentially deceptive news item from FBIS, an automated countermeasure might provide an alert using adaptive fraud detection algorithms [36] or through a retrieval mechanism allow the analyst to quickly assemble and interactively analyze related documents bearing on the potential misinformation. The author is currently developing both of these countermeasures.

Information retrieval, or document retrieval, developed historically to serve the needs of scientists and legal researchers, among others. Despite occasional hoaxes and falsifications of data in these domains, the overwhelming expectation is that documents retrieved are honest representations of attempts to discover scientific truths, or to make a sound legal argument. This assumption does not hold for intelligence and security informatics. Most information retrieval systems are based either on: (a) an exact match Boolean logic by which the system divides the document collection into those documents matching the logic of the request and those that do not, or (b) ranked retrieval. With ranked retrieval a score is derived for each document in the collection based on a measure of similarity between the query and the docu-

ment's representation, as in the vector space model [83], or based on a probability of relevance [63,82].

Although not implemented in existing systems, a utility theoretic approach to information retrieval [20] shows promise for a theory of intelligence and security informatics. In information retrieval predicting relevance is hard enough. Predicting utility, although harder, would be more useful. When information contained in, say, a FBIS document, may be misinformation, then the notion of utility theoretic retrieval, becomes more important. The provider of the content may have believed the information to be true or false, aside from whether it was true or false in some objective sense. The content may be of great value to the intelligence analyst, whether it is true or false, but, in general, it would be important to know not only whether it was true or false, but also whether the provider believed it to be true or false. Current information retrieval algorithms would not take any of these complexities into account in calculating a probability of relevance.

Predictive modeling using the concepts of cognitive hacking and utility-theoretic information retrieval can be applied in two intelligence and security informatics settings which are mirror images of each other, i.e., the user's model of the system's document content and the systems model of the user as a potential malicious insider. Consider an environment where an intelligence analyst accesses sensitive and classified information from intelligence databases. The accessed information itself may represent cognitive attacks coming from the sources from which it has been gathered, e.g., FBIS documents. As discussed above, each of these documents will have a certain utility for the analyst, based on the analyst's situation, based on whether or not the documents contain misinformation, and, if the documents do contain misinformation, whether, or not, the analyst can determine that the misinformation is present. On the other hand, the analyst might be a malicious insider engaged in espionage. The document system will need to have a cost model for each of its documents and will need to build a model of each user, based on the user's transactions with the document system and other external actions.

Denning's theory of information warfare [28] and an information theoretic approach to the value of information [22,23] can be used to rank potential risks given the value of each document held by the system. Particular attention should be paid to deception on the part of the trusted insider to evade detection. Modeling the value of information to adversaries will enable prediction of which documents are likely espionage targets and will enable development of hypotheses for opportunistic periods and scenarios for compromise. These models will be able to detect unauthorized activity and to predict the course of a multi-stage attack so as to inform appropriate defensive actions.

Misinformation, or cognitive hacking, plays a much more prominent role in intelligence and security informatics than it has played in traditional scientific informatics.

The status of content as information, or misinformation, in turn, influences its utility for users. Cognitive hacking countermeasures are needed to detect and defend against cognitive hacking.

2. Examples of Cognitive Hacking

This section summarizes several documented examples of cognitive hacking on the Internet and provides a more detailed discussion of the problem of the insider threat.

2.1 Internet Examples

Table I categorizes the types according to mode and goals. The various modes have been defined above while the goals are self-explanatory and discussed in more detail in the examples themselves.

2.1.1 NEI Webworld Case

In November 1999 two UCLA graduates students and one of their associates purchased almost all of the shares of the bankrupt company NEI Webworld at a price

TABLE I
HACKING WITH THE GOAL OF MODIFYING USER BEHAVIOR

Goals mode	Syntactic	Cognitive overt	Cognitive covert
Denial of Service			
Theft of Services	8		8, 15
Theft of Information			4
Fraud Financial			1, 2, 3, 4, 5
Fraud non-Financial			6, 7
Political		10, 11, 14, 15, 17	17
Commercial or Private Perception Management		6, 9	6
Self-aggrandizement		12, 13, 15	
White Hat Hack		13, 16	

1. NEI Webworld pump and dump.
2. Jonathan Lebed case.
3. Fast-trades.com website pump and dump.
4. PayPal.com.
5. EMULEX press release to Internet Wire.
6. Non-financial fraud—searchengine optimization.
7. Non-financial fraud—CartoonNetwork.com.
8. Bogus virus patch report.
9. Usenet perception management.
10. Hamas site.
11. Ariel Sharon site.
12. New York Times site.
13. Yahoo site.
14. Afghanistan related websites.
15. Fluffi Bunni declares Jihad.
16. CNN site.
17. WTO site.

ranging from $0.05 to $0.17 per share. They opened many Internet message board accounts using a computer at the UCLA BioMedical Library and posted more than 500 messages on hot websites to pump up the stock of the company, stating false information about the company with the purpose of convincing others to buy stock in the company. They claimed that the company was being taken over and that the target price per share was between 5 and 10 dollars. Using other accounts they also pretended to be an imaginary third party, a wireless telecommunications company, interested in acquiring NEI Webworld. What the three men did not post was the fact that NEI was bankrupt and had liquidated assets in May 1999. The stock price rose from $0.13 to $15 in less then one day, and they realized about $364,000 in profits. The men were accused of selling their shares incrementally, setting target prices along the way as the stock rose. On one day the stock opened at $8 and soared to $15 5/16 a share by 9:45 a.m. ET and by 10:14 a.m. ET, when the men no longer had any shares, the stock was worth a mere 25 cents a share.

On Wednesday, December 15, 1999, the US Securities and Exchange Commission (SEC) and the United States Attorney for the Central District of California charged the three men with manipulating the price of NEI Webworld, Inc. In late January 2001, two of them, agreed to gave up their illegal trading profits (approximately $211,000). The Commission also filed a new action naming a fourth individual, as participating in the NEI Webworld and other Internet manipulations. Two of the men were sentenced on January 22, 2001 to 15 months incarceration and 10 months in a community corrections center. In addition to the incarcerations, Judge Feess ordered the men to pay restitution of between $566,000 and $724,000. The judge was to hold a hearing on February 26 to set a specific figure [86]. Anyone with access to a computer can use as many screen names as desired to spread rumors in an effort to pump up stock prices by posting false information about a particular company so that they can dump their own shares and give the impression that their own action has been above board.

2.1.2 The Jonathan Lebed Case

A 15 years old student using only AOL accounts with several fictitious names was able to change the behavior of many people around the world making them act to his advantage [58]. In six months he gained between $12,000 and $74,000 daily each time he posted his messages and, according to the US Security Exchange Commission, he did that 11 times increasing the daily trading volume from 60,000 shares to more that a million. His messages sounded similar to the following one [59]:

```
DATE: 2/03/00 3:43pm Pacific Standard Time
FROM: LebedTG1
```

```
FTEC is starting to break out! Next week, this thing will
EXPLODE...
Currently FTEC is trading for just $21/2. I am expect-
ing to see FTEC at $20
VERYSOON...
Let me explain why...
Revenues for the year should very conservatively be around
$20 million. The average company in the industry trades
with a price/sales ratio of 3.45. With 1.57 million shares
outstanding, this will value FTEC at... $44.
It is very possible that FTEC will see $44, but since I
would like to remain very conservative... my short term
price target on FTEC is still
$20!
The FTEC offices are extremely busy... I am hearing that
a number of HUGE deals are being worked on. Once we get
some news from FTEC and the word gets out about the com-
pany... it will take-off to MUCH HIGHER LEVELS!
I see little risk when purchasing FTEC at these DIRT-CHEAP
PRICES. FTEC is making TREMENDOUS PROFITS and is trad-
ing UNDER BOOK VALUE!!!
This is the #1 INDUSTRY you can POSSIBLY be in RIGHT NOW.
There are thousands of schools nationwide who need FTEC
to install security systems... You can't find a better
positioned company than FTEC!
These prices are GROUND-FLOOR! My prediction is that this
will be the #1 performing stock on the NASDAQ in 2000.
I am loading up with all of the shares of FTEC I possi-
bly can before it makes a run to $20.
Be sure to take the time to do your research on FTEC! You
will probably never come across an opportunity this HUGE
ever again in your entire life.
```

He sent this kind of message after having bought a block of stocks. The purpose was to influence people and let them behave to pump up the price by recommending the stock. The messages looked credible and people did not even think to investigate the source of the messages before making decisions about their money. Jonathan gained $800,000 in six months. Initially the SEC forced him to give up everything, but he fought the ruling and was able to keep part of what he gained. The question is whether he did something wrong, in which case the SEC should have kept everything. The fact that the SEC allowed Jonathan to keep a certain amount of money shows that it is not clear whether or not the teenager is guilty from a legal per-

spective. Certainly, he made people believe that the same message was post by 200 different people.

Richard Walker, the SEC's director of enforcement, referring to similar cases, stated that on the Internet there is no clearly defined border between reliable and unreliable information, investors must exercise extreme caution when they receive investment pitches online.

2.1.3 Fast-Trade.com Website Pump and Dump

In February and March 1999, Douglas Colt, a Georgetown University law student, manipulated four traded stocks using the website Fast-trade.com. Together with a group of friends he posted hundreds of false or misleading messages on Internet message boards such as Yahoo! Finance Raging Bull with the purpose of encouraging people to follow Fast-trade.com advice. The site offered a trial subscription and in less then two months more than 9000 users signed up. The group was able to gain more than $345,000.

2.1.4 PayPal.com

> "We regret to inform you that your username and password have been lost in our database. To help resolve this matter, we request that you supply your login information at the following website."

Many customers of PayPal received this kind of e-mail and subsequently gave personal information about their PayPal account to the site linked by the message (http://paypalsecure.com not http://www.paypal.com) [52]. The alleged perpetrators apparently used their access to PayPal accounts in order to purchase items on eBay.

2.1.5 Emulex Corporation

Mark S. Jakob, after having sold 3000 shares of Emulex Corporation in a "short sale" at prices of $72 and $92, realized that, since the price rose to $100, he lost almost $100,000 [62]. This kind of speculation is realized by borrowing shares from a broker and selling them in hope that the price will fall. Once this happens, the shares are purchased back and the stock is returned to the broker with the short seller keeping the difference.

On August 25th 2000, when he realized the loss, he decided to do something against the company. The easiest and most effective action was to send a false press release to Internet Wire Inc. with the goal of influencing the stock price. He claimed that Emulex Corporation was being investigated by the Security and Exchange Commission (SEC) and that the company was forced to restate 1998 and 1999 earnings. The story quickly spread, and half an hour later other news services such as Dow

Jones, Bloomberg, and CBS Marketwatch picked up the hoax. Due to this false information, in a few hours Emulex Corporation lost over $2 billion dollars. After sending misinformation about the company, Jakob executed trades so that he earned $236,000. Jakob was arrested and charged with disseminating a false press release and with security fraud. He is subject to a maximum of 25 years in prison, a maximum fine of $220 million, two times investor losses, and an order of restitution up to $110 million to the victims of his action.

2.1.6 Non-financial Fraud—Web Search Engine Optimization

Con artists have defrauded consumers for many years over the telephone and via other means of communication, including direct personal interaction. Such financially-motivated fraud continues over the Internet, as described above. Some cognitive hacking uses misinformation in a fraudulent way that does not directly attack the end user.

One such use of misinformation is a practice [61] that has been called "search engine optimization," or "index spamming." Because many users of the Internet find pages through use of Web search engines, owners of websites seek to trick Web search engines to rank their sites more highly when searched by Web search engines. Many techniques, for example, inaccurate metadata, printing white text on white background (invisible to a viewer of the page, but not to a search engine) are used. While this practice does not directly extort money from a user, it does prevent the user from seeing the search results that the user's search would have returned based on the content of the website. Thus the primary attack is on the search engine, but the ultimate target of the attack is the end user. Developers at Web search engines are aware of this practice by website promoters and attempt to defeat it, but it is an on-going skirmish between the two camps.

2.1.7 Non-financial Fraud—CartoonNetwork.com

Another common misinformation practice is to register misleading website names, e.g., a name that might be expected to belong to a known company, or a close variant of it, such as a slight misspelling. In October 2001, the FTC [94] sought to close thousands of websites that allegedly trap Web users after they go to a site with a misleading name. According to the FTC, John Zuccarini registered slight misspelling of hundreds of popular Internet domain names. When a user goes to one of these sites a series of windows advertising various products opens rapidly, despite user attempts to back out of the original site. Zuccarini allegedly made $800,000 to $1,000,000 annually in advertising fees for such attacks.

2.1.8 Bogus Virus Patch Report

Although computer viruses are syntactic attacks, they can be spread through cognitive attacks. The W32/Redesi-B virus [88] is a worm which is spread through Microsoft Outlook. The worm is contained in an e-mail message that comes with a subject chosen randomly from 10 possible subjects, e.g., "FW: Security Update by Microsoft." The text of the e-mail reads "Just received this in my email I have contacted Microsoft and they say it's real" and then provides a forwarded message describing a new e-mail spread virus for which Microsoft has released a security patch which is to be applied by executing the attached file. The attached file is the virus. Thus a virus is spread by tricking the user into taking action thought to prevent the spread of a virus.

2.1.9 Usenet Perception Management

Since the Internet is an open system where everybody can put his or her opinion and data, it is easy to make this kind of attack. Each user is able to influence the whole system or only a part of it in many different ways, for example, by building a personal website or signing up for a newsgroup. Blocking the complete freedom to do these activities, or even checking what people post on the Web, goes against the current philosophy of the system. For this reason technologies for preventing, detecting, and recovering from this kind of attack are difficult to implement [17].

2.1.10 Political Website Defacements—Ariel Sharon Site

Website defacements are usually overt cognitive attacks. For example, in January 2001, during an Israeli election campaign, the website of Likud leader Ariel Sharon was attacked [8]. In this attack, and in the retaliatory attack described in Section 2.1.11, no attempt was made to deceive viewers into thinking that the real site was being viewed. Rather the real site was replaced by another site with an opposing message. The Sharon site had included a service for viewers that allowed them to determine the location of their voting stations. The replacement site had slogans opposing Sharon and praising Palestinians. It also had a feature directing viewers to Hezbollah "polling stations."

2.1.11 Political Website Defacements—Hamas Site

Following the January attack on the Sharon website, the website of the militant group Hamas was attacked in March 2001 [7]. When the Hamas website was hacked, viewers were redirected to a hard-core pornography site.

2.1.12 New York Times Site

In February 2001 the New York Times website was defaced by a hacker identified as "splurge" from a group called "Sm0ked Crew," which had a few days previously defaced sites belonging to Hewlett-Packard, Compaq, and Intel [79,80]. The New York Times defacement included html, a .MID audio file, and graphics. The message stated, among other things, "Well, admin I'm sorry to say by you have just got sm0ked by splurge. Don't be scared though, everything will be all right, first fire your current security advisor…" Rather than being politically motivated, such defacements as these appear to be motivated by self-aggrandizement.

2.1.13 Yahoo Site

In September of 2001 Yahoo's news website was edited by a hacker [68]. This cognitive hacking episode, unlike the defacements discussed above, was more subtle. While not as covert as hacking with the intent to engage in fraud or perception management, neither were the changes made to the website as obvious as those of a typical defacement. A 20-year old researcher confessed that he altered a Reuters news article about Dmitry Skylarov, a hacker facing criminal charges. The altered story stated that Skylarov was facing the death penalty and attributed a false quote to President Bush with respect to the trial.

2.1.14 Website Defacements Since 11 September Terrorist Incident

Since the 11 September terrorist incident, there have been numerous examples of website defacements directed against websites related to Afghanistan [57]. While official Taliban sites have been defaced, often sites in any way linked with Afghanistan were defaced indiscriminately, regardless of which sides they represented in the conflict.

2.1.15 Fluffi Bunni Declares Jihad

Another type of politically motivated cognitive hacking attack has been perpetrated by "Fluffi Bunni," who has redirected numerous websites to a page in which Bunni's opinion on current events is presented. This redirection appears to have been accomplished through a hacking of the Domain Name System Server of NetNames [41].

2.1.16 Website Spoofing—CNN Site

On 7 October 2001, the day that the military campaign against Afghanistan began, the top-ranked news story on CNN's most popular list was a hoax, "Singer Britney Spears Killed in Car Accident." The chain of events which led to this listing started with a website spoofing of http://CNN.com [75]. Then, due to a bug in CNN's software, when people at the spoofed site clicked on the "E-mail This" link, the real CNN system distributed a real CNN e-mail to recipients with a link to the spoofed page. At the same time with each click on "E-mail This" at the bogus site, the real site's tally of most popular stories was incremented for the bogus story. Allegedly this hoax was started by a researcher who sent the spoofed story to three users of AOL's Instant Messenger chat software. Within 12 h more than 150,000 people had viewed the spoofed page.

In 1997 Felton and his colleagues showed that very realistic website spoofings could be readily made [37]. More recently, Yuan et al. [97] showed that these types of website spoofs could be done just as easily with more contemporary Web technologies.

2.1.17 Website Spoofing—WTO Site

Use of misleading domain names can also be political and more covert. Since 1999, a site, http://www.gatt.org, has existed which is a parody of the World Trade Organization site, http://www.wto.org [73]. Again, as in the case of the spoofing of the Yahoo new site mentioned above, the parody can be seen through fairly easily, but still could mislead some viewers.

2.2 Insider Threat

Trusted insiders who have historically caused the most damage to national security were caught only after prolonged counterintelligence operations. These insiders carried out their illegal activities for many years without raising suspicion. Even when it was evident that an insider was misusing information, and even when attention began to focus on the insider in question as a suspect, it took more years before the insider was caught. Traditionally apprehension of trusted insiders has been possible only after events in the outside world had taken place, e.g., a high rate of double agents being apprehended and executed that led to an analysis eventually focusing on the insider. Once it was clear that there was likely a problem with insider misuse of information, it was eventually possible to determine the identity of the insider by considering who had access to the information and by considering other factors such as results of polygraph tests.

The insider threat, is much more pervasive, however, than a small number of high profile national security cases. It has been estimated that the majority of all computer security breeches are due to insider attacks, rather than to external hacking [4].

As organizations move to more and more automated information processing environments, it becomes potentially possible to detect signs of insider misuse much earlier than has previously been possible. Information systems can be instrumented to record all uses of the system, down to the monitoring of individual keystrokes and mouse movements. Commercial organizations have made use of such clickstream mining, as well as analysis of transactions to build profiles of individual users. Credit card companies build models of individuals' purchase patterns to detect fraudulent usage. Companies such as Amazon.com analyze purchase behavior of individual users to make recommendations for the purchase of additional products, likely to match the individual user's profile.

A technologically adept insider, however, may be aware of countermeasures deployed against him, or her, and operate in such a way as to neutralize the countermeasures. In other words, an insider can engage in cognitive hacking against the network and system administrators. A similar situation arises with Web search engines, where what has been referred to as a cold war exists between Web search engines and search engine optimizers, i.e., marketers who manipulate Web search engine rankings on behalf of their clients.

Models of insiders can be built based on:

(a) known past examples of insider misuse;
(b) the insider's work role in the organization;
(c) the insider's transactions with the information system; and
(d) the content of the insider's work product.

This approach to the analysis of the behavior of the insider is analogous to that suggested for analyzing the behavior of software programs by Munson and Wimer [70]. One aspect of this approach is to look for known signatures of insider misuse, or for anomalies in each of the behavioral models individually. Another aspect is to look for discrepancies among the models. For example, if an insider is disguising the true intent of his, or her, transactions by making deceptive transactions that disguise the true nature of what the insider is doing, then this cognitive hacking might be uncovered by comparing the transactions to the other models described above, e.g., to the insider's work product.

User models have long been of interest to researchers in artificial intelligence and in information retrieval [81,26,49]. Several on-going research programs have been actively involved in user modeling for information retrieval. The Language Modeling approach to probabilistic information retrieval has begun to consider query (user) models [53,54]. The Haystack project at MIT is building models of users based on

their interactions with a document retrieval system and the user's collections of documents. The current focus of this project, however, seems to be more on overall system architecture issues, rather than on user modeling as such [46].

The current type of user modeling that might provide the best basis for cognitive hacking countermeasures is recommender system technology [92,93,44]. One of the themes of the recommender systems workshop held at the 1999 SIGIR conference [43] was the concern to make recommender systems applicable to problems of more importance than selling products. Since then, recommender systems technology has developed, but applications are generally still largely commercial. Researchers are concerned with developing techniques that work well with sparse amounts of data [31] and with scaling up to searching tens of millions of potential neighbors, as opposed to the tens of thousands of today's commercial systems [84]. Related to this type of user modeling, Anderson and Khattak [5] described preliminary results with the use of an information retrieval system to query an indexed audit trail database, but this work was never completed [3].

3. Economic and Digital Government Issues Related to Cognitive Hacking

3.1 An Information Theoretic Model of Cognitive Hacking

Information theory has been used to analyze the value of information in horse races and in optimal portfolio strategies for the stock market [22]. We have begun to investigate the applicability of this analysis to cognitive hacking. So far we have considered the simplest case, that of a horse race with two horses. But the analysis can be easily extended to the case of the stock market.

Sophisticated hackers can use information theoretic models of a system to define a gain function and conduct a sensitivity analysis of its parameters. The idea is to identify and target the most sensitive variables of the system, since even slight alterations of their value may influence people's behavior. For example, specific information on the health of a company might help stock brokers predict fluctuations in the value of its shares. A cognitive hacker manipulates the victim's perception of the likelihood of winning a high payoff in a game. Once the victim has decided to play, the cognitive hacker influences which strategy the victim chooses.

3.1.1 A Horse Race

Here is a simple model illustrating this kind of exploit. A horse race is a system defined by the following elements [22]:

- There are m horses running in a race;
- Each horse i is assigned a probability p_i of winning the race (so $\{p_i\}$, $i = 1, \ldots, m$ is a probability distribution);
- Each horse i is assigned an odds o_i signifying that a gambler that bet b_i dollars on horse i would win $b_i o_i$ dollars in case of victory (and suffer a total loss in case of defeat).

If we consider a sequence of n independent races, it can be shown that the average rate of the wealth gained at each race is given by

$$W(b, p, o) = \sum_{i=1}^{m} p_i \log b_i o_i,$$

where b_i is the percentage of the available wealth invested on horse i at each race. So the betting strategy that maximizes the total wealth gained is obtained by solving the following optimization problem:

$$W(p, o) = \max_b W(b, p, o) = \max_b \sum_{i=1}^{m} p_i \log b_i o_i$$

subject to the constraint that the b_i's add up to 1. It can be shown that this solution turns out to be simply $b = p$ (proportional betting) and so $W(p, o) = \sum_{i=1}^{m} p_i \log p_i o_i$.

Thus, a hacker can predict the strategy of a systematic gambler and make an attack with the goal of deluding the gambler on his/her future gains. For example, a hacker might lure an indecisive gambler to invest money on false prospects. In this case it would be useful to understand how sensitive the function W is to p and o and tamper with the data in order to convince a gambler that it is worth playing (because W appears illusionary larger than it actually is).

To study the sensitivity of W to its domain variables we consider the partial derivatives of W with respect to p_i and o_i and see where they assume the highest values. This gives us information on how steep the function W is on subsets of its domain.

If we consider the special case of races involving only two horses ($m = 2$), then we have

$$W(p, o_1, o_2) = p \log po_1 + (1 - p) \log(1 - p)o_2,$$

- $\frac{\partial W}{\partial p}(p, o_1, o_2) = \log\left(\frac{p}{1-p}\frac{o_1}{o_2}\right)$,
- $\frac{\partial W}{\partial o_1}(p, o_1, o_2) = \frac{p}{o_1}$,
- $\frac{\partial W}{\partial o_2}(p, o_1, o_2) = \frac{1-p}{o_2}$.

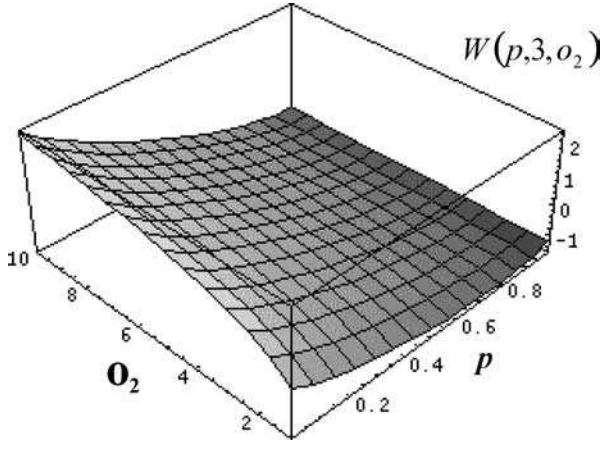


FIG. 2.

Thus, if we fix one of the variables then we can conduct a graphic analysis of those functions with a 3D plot, see Fig. 2.

CASE 1. o_1 is constant. This is the doubling rate function. The most sensitive parameter to let W increase is o_2. Increasing this variable W grows at a fast rate for low values of p and grows with a smaller rate for higher values of p.

3.1.2 Applying the Horse Race Example to the Internet

Let's take into consideration the Mark Jakob case discussed earlier. In this example the two horses are: horse 1, Emulex stock goes up; and horse 2, Emulex stock goes down. First, the cognitive hacker makes the victim want to play the game by making the victim think that he can make a large profit through Emulex stock transactions. This is done by spreading misinformation about Emulex, whether positive or negative, but news that, if true would likely cause the stock's value to either sharply increase, or decrease, respectively. Positive misinformation might be the news that Emulex had just been granted a patent that could lead to a cure for AIDS. Negative misinformation might be that Emulex was being investigated by the Securities and Exchange Commission (SEC) and that the company was forced to restate 1998 and 1999 earnings. This fraudulent negative information was in fact posted by Jakob.

3.2 Theories of the Firm and Cognitive Hacking

Much attention in economics has been devoted to theories of the market. The economic actor has been modeled as enjoying perfect, costless information. Such analyses, however, are not adequate to explain the operation of firms. Theories of the firm provide a complementary economic analysis taking into account transaction and organizing costs, hierarchies, and other factors left out of idealized market models. It has been argued that information technology will transform the firm, such that "... the fundamental building blocks of the new economy will one day be 'virtual firms,' ever-changing networks of subcontractors and freelancers, managed by a core of people with a good idea" [33]. Others argue that more efficient information flows not only lower transaction costs, thereby encouraging more outsourcing, but also lower organization costs, thereby encouraging the growth of larger companies [2]. More efficient information flow implies a more standardized, automated processing of information, which is susceptible to cognitive attack. In this light Libicki's characterization of semantic attacks in terms of misinformation being inserted into interactions among intelligent software agents [60] can be applied to misinformation's potential to disrupt increasingly automated business processes, as well.

3.3 Digital Government and Cognitive Hacking

The National Center for Digital Government is exploring issues related to the transition from traditional person-to-person provision of government services to the provision of such services over the Internet. As excerpted from the Center's mission statement:

> Government has entered a period of deep transformation heralded by rapid developments in information technologies. The promise of digital government lies in the potential of the Internet to connect government actors and the public in entirely new ways. The outcomes of fundamentally new modes of coordination, control, and communication in government offer great benefits and equally great peril [71].

A digital government workshop held in 2003 [72], focused on five scenarios for future authentication policies with respect to digital identity:

- Adoption of a single national identifier;
- Sets of attributes;
- Business as usual, i.e., continuing growth of the use of ad hoc identifiers;
- Ubiquitous anonymity;
- Ubiquitous identify theft.

The underlying technologies considered for authentication were: biometrics; cryptography, with a focus on digital signatures; secure processing/computation; and reputation systems.

Most of the discussion at the workshop focused on issues related to authentication of users of digital government, but, as the scenario related to ubiquitous identity theft implies, there was also consideration of problems related to misinformation, including cognitive hacking.

In the face-to-face interaction with other people associated with traditional provision of government services, there is normally some context in which to evaluate the reliability of information being conveyed. As we have seen, this type of evaluation cannot be directly transferred to digital government. The Internet's open nature makes it an ideal arena for dissemination of misinformation. What happens if a user makes a decision based on information found on the Web that turns out to be misinformation, even if the information appears to come from a government website? In reality, the information might be coming from a spoofed version of a government website. Furthermore, the insider threat is a serious concern for digital government.

4. Legal Issues Related to Cognitive Hacking

The Internet is a medium of communication, perception management, advertising and e-commerce, among other things. Laws that apply to other media related to these activities also apply to the Internet. However, as stated by Mensik and Fresen [66] of Baker and McKenzie:

> As a new medium, the Internet compels the legal system to reexamine its fundamental principles. To accommodate the new medium, the legal system must adapt traditional doctrines and adopt new concepts in a way that protects rights and establishes a fair framework for assessing liability. Some say that neither common law nor civil law jurisdictions are capable of keeping pace with the rapid development of new technologies. This may be especially true in the global business arena of the Internet that defies the application of traditional legal principles... future electronic transactions will be performed solely between computers by electronic agents or "know-bots" and this lack of human involvement challenges fundamental assumptions of contract law, such as presented in the Uniform Commercial Code, that are necessarily based upon human interaction in the contracting process.

From a cognitive hacking perspective this last point about electronic agents is of interest as it relates this paper's definition of cognitive hacking as being targeted against a human user, to Libicki's [60] original definition of semantic attacks as being against agents (see below).

An attorney from Buchanan Ingersoll P.C. [11] provides this additional perspective:

> Content providers, the businesses which hire them, as well as their advertising agencies and Web site developers, may be liable directly to victims for false or misleading advertising, copyright and trademark infringement, and possibly for vicarious infringements. The difference between other forms of media and the Internet is that the Internet contemplates the publishing and linking of one site to another, which geometrically multiplies the possibility for exposure to claims. The electronic benefits of the Internet promotes the linking of information contained in one site to another and creates a higher likelihood of dissemination of offending material.

The Lanham Act, 15 U.S.C. §1125(a) [65] has been applied to the prosecution of false advertising on the Web. It provides that any person who "... uses in commerce any word, term, name, symbol, or device... false description of origin, false or misleading description of fact... which, (A) is likely to cause confusion, or to cause mistake, or to deceive as to affiliation, connection, or association of such person with another person, or as to the origin, sponsorship, or approval of his or her goods, services, or commercial activities by another person, or (B) in commercial advertising or promotion misrepresents the nature, characteristics, qualities, or geographic origin of his or her or another person's goods, services, or commercial activities, shall be liable in a civil action by any person who believes that he or she is likely to be damaged by such an act."

The Lanham Act, copyright, and trademark law, among other established areas of the law, are being used to decide cases related to cognitive hacking. For example, in the area of search engine optimization, if company A's website uses the trademark of company B in metatags for company A's website in order to divert Web searches for company B to company A's website, then company A could be liable for trademark infringement or false advertising under the Lanham Act. On the other hand, company A could argue that its use of metatags was protected under trademark principles of fair use or First Amendment principles of free speech. As a more extreme action, company A might download the entire content of a popular website from company B and incorporate it into company A's website with all of the company B content printed white-on-white background, so that it would be invisible to human viewers, but visible to Web search engines. This would be a violation of copyright laws and possibly also be considered unfair competition and trademark dilution [1].

The application of the law to cognitive hacking and other areas related to the Internet is a very volatile area of the law. The events of September 2001 have only made the situation more volatile as the debate between privacy and security has shifted. It is to be expected that more legislation affecting this area will be enacted and that the associated case law will continue to evolve over the coming years.

If cognitive hacking challenges security experts because it aims at the vulnerable human/computer interface, it challenges legal experts because it both targets and exploits information, the very lifeblood of free speech and democracy [25]. Criminal prosecution and civil lawsuits may help combat cognitive hacking, but this will be possible only in compliance with free speech protection. Laws already exist that can fight disinformation without violating fundamental rights. But cognitive hacking introduces new characteristics requiring revised legal doctrines. Ultimately, confronting cognitive hacking will require integrating legal and technological anti-hacking tools.

Within the United States, where the US Constitution prevails, legal action seeking to regulate cognitive hacking can conflict with First Amendment free speech protection. The First Amendment prohibits government punishment of "false" ideas or opinions. The competition of ideas, not legislatures or courts, determines what ideas and opinions society accepts. And while false *facts* lack constitutional protection, the US Supreme Court has ruled that the news media as well as non-news media discussion of public issues or persons require some margin of factual error to prevent chilling legitimate debate. "The First Amendment requires that we protect some falsehood in order to protect speech that matters." [39]

Punishing Web defacement should present no First Amendment concerns. As with graffiti in the physical world [74], the First Amendment does not shield unauthorized damage to property of others. Distribution of false information with an intent to defraud or manipulate should also require little First Amendment consideration.

However, a regulatory measure aimed at the content of the cognitive hacking, on what the hacking *says*, would likely fail constitutionally unless the state interest were compelling and no content-neutral alternative were available. Attempts within the United States to stop a website or metatags or list server postings from expressing unpopular views would also struggle to pass First Amendment muster [78].

Indeed, many legal avenues already exist for attacking cognitive hacking consistent with First Amendment rights.

Some forms of cognitive hacking are tightly coupled with conduct that has legal consequences. Web defacement requires breaking into another's website, and, therefore, could be characterized as vandalism, destruction of property, trespassing, or, under the right circumstances, a violation of the Electronic Communications Privacy Act of 1986. As described above, manipulating securities markets through cognitive hacking can trigger prosecution under the securities laws. If unauthorized use of a software robot to harvest information from another's website can be trespassing [32], perhaps so is feeding false data into another's software robot that is harvesting Web information.

Other forms of cognitive hacking involve incendiary or factually false statements beyond First Amendment protection. Disseminating false information to secure an

agreement could be the basis of legal actions for fraud or misrepresentation. Disseminating false information that damages the reputation of a person, business, or product could lead to a libel, defamation, or commercial disparagement suit. Incorporating trademarks of others as metatags that mislead consumers about the origin of goods or services or reduce the goodwill associated with a mark could be reached through the legal remedies provided by trademark and trademark antidilution statutes.

The special persuasive powers of computer output create an extra dimension of legal concern. Humans are quick to believe what they read and see on their computer screen. Even today, it is common to hear someone say a fact must be true because they read it on the Web. A website's anthropomorphic software agent is likely to enjoy greater credibility than a human, yet no conscience will prevent an anthropomorphic agent from saying whatever it has been programmed to say [42]. Cognitive hackers may, therefore, require new legal doctrines because their mechanisms apparently bypass normal human critical thinking.

Still more elusive will be identifying and taking meaningful legal action against the perpetrator. The speed and lack of human intervention that is typically associated with cognitive hacking, combined with the lack of definitive identification information generally inherent in the Internet's present architecture, complicate legal proof of who is the correct culprit. Privacy protection makes the task more difficult. Even if identified, the individual or entity may disappear or lack the assets to pay fines or damages.

Attention may, therefore, focus instead on third-party intermediaries, such as Internet service providers, websites, search engines, and so forth, just as it has for Internet libel, copyright infringement, and pornography. Intermediaries are likely to have greater visibility and more assets, making legal action easier and more productive. A cognitive hacking victim might contend that an intermediary or the producer of Web-related software failed to take reasonable measures to defend against cognitive hacking. An intermediary's legal responsibility will grow as the technological means for blocking cognitive hacking become more effective and affordable. Rapid technological advances with respect to anti-hacking tools would empower raising the bar for what it considered reasonable care.

The actual application of the law to cognitive hacking is still in formation. It is to be expected that case law with respect to cognitive hacking will continue to evolve over the coming years. Enactment of specific legislation is also possible.

5. Cognitive Hacking Countermeasures

Given the variety of approaches documented above and the very nature of cognitive hacking, *preventing* cognitive hacking reduces either to preventing unauthorized

access to information assets (such as in Web defacements) in the first place or detecting posted misinformation before user behavior is affected (that is, before behavior is changed but possibly after the misinformation has been disseminated). The latter may not involve unauthorized access to information, as, for instance, in "pump and dump" schemes that use newsgroups and chat rooms. By definition, *detecting* a successful cognitive hack would involve detecting that the user behavior has already been changed. We are not considering detection in that sense at this time.

Our discussion of methods for preventing cognitive hacking will be restricted to approaches that could automatically alert users of problems with their information source or sources (information on a Web page, newsgroup, chat room, and so on). Techniques for preventing unauthorized access to information assets fall under the general category of computer and network security and will not be considered here. Similarly, detecting that users have already modified their behaviors as a result of the misinformation, namely, that a cognitive hack has been successful, can be reduced to detecting misinformation and correlating it with user behavior.

The cognitive hacking countermeasures discussed here will be primarily mathematical and linguistic in nature. The use of linguistic techniques in computer security has been pioneered by Raskin and colleagues at Purdue University's Center for Education and Research in Information Assurance and Security [6]. Their work, however, has not addressed cognitive hacking countermeasures.

5.1 Single Source Cognitive Hacking

In this section, we propose a few possible approaches for the single source problem. By single source, we mean situations in which redundant, independent sources of information about the same topic are not available. An authoritative corporate personnel database would be an example.

5.1.1 Authentication of Source

This technique involves due diligence in authenticating the information source and ascertaining its reliability. Various relatively mature certification and PKI technologies can be used to detect spoofing of an information server. Additionally, reliability metrics can be established for an information server or service by scoring its accuracy over repeated trials and different users. In this spirit, Lynch [61] describes a framework in which trust can be established on an individual user basis based on both the identity of a source of information, through PKI techniques for example, and in the behavior of the source, such as could be determined through rating systems. Such an approach will take time and social or corporate consensus to evolve.

5.1.2 Information "Trajectory" Modeling

This approach requires building a model of a source based on statistical historical data or some sort of analytic understanding of how the information relates to the real world. For example, weather data coming from a single source (website or environmental sensor) could be calibrated against historical database (from previous years) or predictive model (extrapolating from previous measurements). A large deviation would give reason for hesitation before committing to a behavior or response.

As an interesting aside, consider the story lines of many well-scripted mystery novels or films. We believe that the most satisfying and successful stories involve s sequence of small deviations from what is expected. Each twist in the story is believable but when aggregated, the reader or viewer has reached a conclusion quite far from the truth. In the context of cognitive hacking, this is achieved by making a sequence of small deviations from the truth, not one of which fails a credibility test on it own. The accumulated deviations are, however, significant and surprise the reader or viewer who was not paying much attention to the small deviations one by one. However, a small number of major "leaps of faith" would be noticed and such stories are typically not very satisfying. Modeling information sources is something that can be done on a case-by-case basis as determined by the availability of historical data and the suitability of analytic modeling.

5.1.3 Ulam Games

Stanislaw Ulam in his autobiography *Adventure of a Mathematician* posed the following question [91]:

> "Someone thinks of a number between one and one million (which is just less than 2^{20}). Another person is allowed to ask up to twenty questions, to which the first person is supposed to answer only yes or no. Obviously, the number can be guessed by asking first: "Is the number in the first half-million?" and then again reduce the reservoir of numbers in the next question by one-half, and so on. Finally, the number is obtained in less than $\log_2(1000000)$. Now suppose one were allowed to lie once or twice, then how many questions would one need to get the right answer?"

Of course, if an unbounded number of lies are allowed, no finite number of questions can determine the truth. On the other hand, if say k lies are allowed, each binary search question can be repeatedly asked $2k + 1$ times which is easily seen to be extremely inefficient. Several researchers have investigated this problem, using ideas from error-correcting codes and other areas [18,69].

This framework involves a sequence of questions and a bounded number of lies, known a priori. For these reasons, we suspect that this kind of model and solution approach may not be useful in dealing with the kinds of cognitive hacking we have

documented, although it will clearly be useful in cognitive hacking applications that involve a sequence of interactions between a user and an information service, as in a negotiation or multi-stage handshake protocol.

5.1.4 Linguistic Countermeasures with Single Sources

5.1.4.1 Genre Detection and Authority Analysis. A careful human reader of some types of misinformation, e.g., exaggerated pump and dump scheme postings on the Web about a company's expected stock performance, can often detect the misinforming posting from other legitimate postings, even if these legitimate postings are also somewhat hyperbolic. Since Mosteller and Wallace's seminal work on authorship attribution in the 1964 [67], statistical linguistics approaches have been used to recognize the style of different writings. In Mosteller and Wallace's work this stylistic analysis was done to determine the true author of anonymous Federalist papers, where the authorship was disputed. Since then Biber and others [9,10,35, 50] have analyzed the register and genre of linguistic corpora using similar stylistic analysis. Kessler et al. [51] have developed and tested algorithms based on this work to automatically detect the genre of text.

5.1.4.2 Psychological Deception Cues. The approach to genre analysis taken, e.g., by Biber and Kessler et al., is within the framework of corpus linguistics, i.e., based on a statistical analysis of general word usage in large bodies of text. The work on deception detection in the psychology and communications fields (see Section 1.5) is based on a more fine-grained analysis of linguistic features or cues. Psychological experiments have been conducted to determine which cues are indicative of deception. To date this work has not led to the development of software tools to automatically detect deception in computer-mediated communication, but researchers see the development of such tools as one of the next steps in this line of research [99].

5.2 Multiple Source Cognitive Hacking

In this section, we discuss possible approaches to preventing cognitive hacking when multiple, presumably redundant, sources of information are available about the same subject of interest. This is clearly the case with financial, political and other types of current event news coverage.

Several aspects of information dissemination through digital, network media, such as the Internet and World Wide Web, make cognitive hacking possible and, in fact, relatively easy to perform. Obviously, there are enormous market pressures on the news media and on newsgroups to quickly disseminate as much information as possible. In the area of financial news, in particular, competing news services strive to

be to the first to give reliable news about breaking stories that impact the business environment. Such pressures are at odds with the time consuming process of verifying accuracy. A compromise between the need to quickly disseminate information and the need to investigate its accuracy is not easy to achieve in general.

Automated software tools could in principle help people make decisions about the veracity of information they obtain from multiple networked information systems. A discussion of such tools, which could operate at high speeds compared with human analysis, follows.

5.2.1 Source Reliability via Collaborative Filtering and Reliability Reporting

The problem of detecting misinformation on the Internet is much like that of detecting other forms of misinformation, for example, in newsprint or verbal discussion. Reliability, redundancy, pedigree and authenticity of the information being considered are key indicators of the overall "trustworthiness" of the information. The technologies of collaborative filtering and reputation reporting mechanisms have been receiving more attention recently, especially in the area of on-line retail sales [96,27]. This is commonly used by the many on-line price comparison services to inform potential customers about vendor reliability. The reliability rating is computed from customer reports.

Another technology, closely related to reliability reporting is collaborative filtering. This can be useful in cognitive hacking situations that involve opinions rather than hard objective facts. See [90] for details about collaborative filtering approaches.

Both of these approaches involve user feedback about information that they receive from a particular information service, building up a community notion of reliability and usefulness of a resource. The automation in this case is in the processing of the user feedback, not the evaluation of the actual information itself.

5.2.1.1 An Example of a Multiple Source Collaborative Filtering Model for Multiple News Sources.

Consider the following scenario. An end user is examining a posting to the business section of Google News [40]. The document purports to provide valuable news about a publicly traded company that the user would like to act on quickly by purchasing, or selling stock. Although this news item might be reliable, it might also be misinformation being fed to unwary users by a cognitive hacker as part of a pump and dump scheme, i.e., a cognitive hacker's hyping of a company by the spread of false, or misleading information about the company and the hacker's subsequent selling of the stock as the price of its shares rise, due to the misinformation. The end user would like to act quickly to optimize

his or her gains, but could pay a heavy price, if this quick action is taken based on misinformation.

A cognitive hacking countermeasure is under development which will allow an end user to effectively retrieve and analyze documents from the Web that are similar to the original news item. First, a set of documents retrieved by the Google News clustering algorithm. The Google News ranking of the clustered documents is generic, not necessarily optimized as a countermeasure for cognitive attacks. We are developing a combination process in which several different search engines are used to provide alternative rankings of the documents initially retrieved by Google News. The ranked lists from each of these search engines, along with the original ranking from Google News, will be combined using the Combination of Expert Opinion algorithm [64] to provide a more optimal ranking. Relevance feedback judgments from the end user will be used to train the constituent search engines. It is expected that this combination and training process will yield a better ranking than the initial Google News ranking. This is an important feature in a countermeasure for cognitive hacking, because a victim of cognitive hacking will want to detect misinformation as soon as possible in real time.

5.2.2 Byzantine Generals Models

Chandy and Misra [15] define the Byzantine General's Problem as follows:

> A message-communicating system has two kinds of processes, *reliable* and *unreliable*. There is a process, called *general*, that may or may not be reliable. Each process x has a local variable $byz[x]$. It is required to design an algorithm, to be followed by all reliable processes, such that every reliable process x eventually sets its local variable $byz[x]$, to a common value. Furthermore, if *general* is reliable, this common value is $d0[g]$, the initial value of one of *general*'s variables. The solution is complicated by the fact that unreliable processes send arbitrary messages. Since reliable processes cannot be distinguished from the unreliable ones, the straightforward algorithm--*general* transmits its initial value to all processes and every reliable process u assigns this value to $byz[u]$--does not work, because *general* itself may be unreliable, and hence may transmit different values to different processes.

This problem models a group of generals plotting a coup. Some generals are reliable and intend to go through with the conspiracy while others are feigning support

and in fact will support the incumbent ruler when the action starts. The problem is to determine which generals are reliable and which are not.

Just as with the Ulam game model for a single information source, this model assumes a sequence of interactions according to a protocol, something that is not presently applicable to the cognitive hacking examples we have considered, although this model is clearly relevant to the more sophisticated information sources that might arise in the future.

5.2.3 Detection of Collusion by Information Sources

Collusion between multiple information sources can take several forms. In pump and dump schemes, a group may hatch a scheme and agree to post-misleading stories on several websites and newsgroups. In this case, several people are posting information that will have common facts or opinions, typically in contradiction to the consensus.

Automated tools for preventing this form of cognitive hack would require natural language processing to extract the meaning of the various available information sources and then compare their statistical distributions in some way. For example, in stock market discussion groups, a tool would try to estimate the "position" of a poster, from "strong buy" to "strong sell" and a variety of gradations in between. Some sort of averaging or weighting could be applied to the various positions to determine a "mean" or expected value, flagging large deviations from that expected value as suspicious.

Similarly, the tool could look for tightly clustered groups of messages, which would suggest some form of collusion. Such a group might be posted by the one person or by a group in collusion, having agreed to the form of a cognitive hack beforehand.

Interestingly, there are many statistical tests for detecting outliers but much less is known about detecting collusion which may not be manifest in outliers but in unlikely clusters that may not be outliers at all. For example, if too many eyewitnesses agree to very specific details of a suspect's appearance (height, weight, and so on), this might suggest collusion to an investigator. For some interesting technology dealing with corporate insider threats due to collusion, see [100].

Automated software tools that can do natural language analysis of multiple documents, extract some quantitative representation of a "position" based on that document and then perform some sort of statistical analysis of the representations are in principle possible, but we are not aware of any efforts working at developing such a capability at this time.

5.2.4 Linguistic Countermeasures with Multiple Sources

5.2.4.1 Authorship Attribution. Stylistic techniques from linguistics are also potential tools for determining the likelihood of authenticity of multiple documents being analyzed [67]. Suppose we are given a set of documents authored by one or more people hiding themselves under possibly multiple pseudonyms. It would be desirable to group the documents according to the real author; that is, to partition the documents into subsets of papers all belonging to the same author.

The main idea is to embed the given document into a finite-dimensional linear feature space of stylistic language usage with some notion of stylistic distance in that space. By performing cluster and other types of analyses on the writing and linguistic style of the whole document or sections thereof, it might be possible to establish which sections of documents are stylistically similar and so, presumably, authored by the same writer.

This kind of detection cannot be applied to short messages, but for a consistent length and enough words, it could determine, with high confidence the stylistic characteristic of the author, or source [77].

6. Future Work

In this chapter a variety of cognitive hacking countermeasures have been described, but implementation has begun on only a few of them. Our future work lies in implementation of the remaining countermeasures and in the development of countermeasures that can be used not only against cognitive attacks, but against semantic attacks more broadly, such as the attacks with misinformation against autonomous agents, as described in Libicki's original definition of semantic hacking.

7. Summary and Conclusions

This chapter has defined a new concept in computer network security, cognitive hacking. Cognitive hacking is related to other concepts, such as semantic hacking, information warfare, and persuasive technologies, but is unique in its focus on attacks via a computer network against the mind of a user. Psychology and Communications researchers have investigated the closely related area of deception and detection in interpersonal communication, but have not yet begun to develop automated countermeasures. We have argued that cognitive hacking is one of the main features which distinguishes intelligence and security informatics from traditional scientific, medical, or legal informatics. If, as claimed by psychologists studying interpersonal

deception, most interpersonal communication involves some level of deception, then perhaps communication via the Internet exhibits a level of deception somewhere between that of face-to-face interpersonal communication, on the one hand, and scientific communication on the other. As the examples from this chapter show, the level of deception on the Internet and in other computer networked settings is significant, and the economic losses due to cognitive hacking are substantial. The development of countermeasures against cognitive hacking is an important priority.

ACKNOWLEDGEMENTS

Support for this research was provided by a Department of Defense Critical Infrastructure Protection Fellowship grant with the Air Force Office of Scientific Research, F49620-01-1-0272; Defense Advanced Research Projects Agency Projects F30602-00-2-0585 and F30602-98-2-0107; and the Office of Justice Programs, National Institute of Justice, Department of Justice Award 2000-DT-CX-K001 (S-1). The views in this document are those of the authors and do not necessarily represent the official position of the sponsoring agencies or of the US Government.

REFERENCES

[1] Abel S., "Trademark issues in cyberspace: The brave new frontier", http://library.lp.findlaw.com/scripts/getfile.pl?file=/firms/fenwick/fw000023.html, 1998.
[2] Agre P., "The market logic of information", *Knowledge, Technology, and Policy* **13** (1) (2001) 67–77.
[3] Anderson R., Personal communication, 2002.
[4] Anderson R.H., Bozek T., Longstaff T., Meitzler W., Skroch M., Van Wyk K., "Research on mitigating the insider threat to information systems – #2", in: *Proceedings of a Workshop Held August 2000. RAND Technical Report CF163*, RAND, Santa Monica, CA, 2000.
[5] Anderson R., Khattak A., "The use of information retrieval techniques for intrusion detection", in: *First International Workshop on Recent Advances in Intrusion Detection (RAID), Louvain-la-Neuve, Belgium*, 1998.
[6] Atallah M.J., McDonough C.J., Raskin V., Nirenburg S., "Natural language processing for information assurance and security: An overview and implementations", in: *Proceedings of the 2000 Workshop on New Security Paradigms*, 2001.
[7] BBC News Online, "Hamas hit by porn attack", http://news.bbc.co.uk/low/english/world/middle_east/newsid_1207000/1207551.stm, 2001.
[8] BBC News Online, "Sharon's website hacked", http://news.bbc.co.uk/low/english/world/middle_east/newsid_1146000/1146436.stm, 2001.
[9] Biber D., *Dimensions of Register Variation: A Cross-Linguistic Comparison*, Cambridge Univ. Press, Cambridge, UK, 1995.

[10] Biber D., "Spoken and written textual dimensions in English: Resolving the contradictory findings", *Language* **62** (2) (1986) 384–413.
[11] Buchanan Ingersoll P.C., "Avoiding web site liability—online and on the hook?", http://library.lp.findlaw.com/scripts/getfile.pl?file=/articles/bipc/bipc000056.html, 2001.
[12] Buller D.B., Burgoon J.K., "Interpersonal deception theory", *Communication Theory* **6** (3) (1996) 203–242.
[13] Burgoon J.K., Blair J.P., Qin T., Nunamaker J.F., "Detecting deception through linguistic analysis", in: *NSF/NIJ Symposium on Intelligence and Security Informatics, June 1–3, 2003, Tucson, AZ*, in: *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2003, pp. 91–101.
[14] Cao J., Crews J.M., Lin M., Burgoon J.K., Nunamaker J.F., "Designing Agent99 trainer: a learner-centered, Web-based training system for deception detection", in: *NSF/NIJ Symposium on Intelligence and Security Informatics, June 1–3, 2003, Tucson, AZ*, in: *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2003, pp. 358–365.
[15] Chandy K.M., Misra J., *Parallel Program Design: A Foundation*, Addison–Wesley, Reading, MA, 1988.
[16] Chen H., Zeng D.D., Schroeder J., Miranda R., Demchak C., Madhusudan T. (Eds.), *Intelligence and Security Informatics: First NSF/NIJ Symposium ISI 2003, Tucson, AZ, June 2003, Proceedings*, Springer-Verlag, Berlin, 2003.
[17] Chez.com, "Disinformation on the Internet", http://www.chez.com/loran/art_danger/art_danger_on_internet.htm, 1997.
[18] Cignoli R.L.O., D'Ottaviano I.M.L., Mundici D., *Algebraic Foundations of Many-Valued Reasoning*, Kluwer Academic, Boston, 1999.
[19] Combs J.E., Nimmo D., *The New Propaganda: The Dictatorship of Palaver in Contemporary Politics*, Longman, New York, 1993.
[20] Cooper W.S., Maron M.E., "Foundations of probabilistic and utility-theoretic indexing", *Journal of the Association for Computing Machinery* **25** (1) (1978) 67–80.
[21] Cornetto K.M., "Identity and illusion on the Internet: Interpersonal deception and detection in interactive Internet environments", PhD thesis, University of Texas at Austin, 2001.
[22] Cover T.A., Thomas J.A., *Elements of Information Theory*, Wiley, New York, 1991.
[23] Cybenko G., Giani A., Thompson P., "Cognitive hacking and the value of information", in: *Workshop on Economics and Information Security, May 16–17, Berkeley, CA*, 2002.
[24] Cybenko G., Giani A., Thompson P., "Cognitive hacking: A battle for the mind", *IEEE Computer* **35** (8) (2002) 50–56.
[25] Cybenko G., Giani A., Heckman C., Thompson P., "Cognitive hacking: Technological and legal issues", in: *LawTech 2002, November 7–9*, 2002.
[26] Daniels P.J., Brooks H.M., Belkin N.J., "Using problem structures for driving human–computer dialogues", in: Sparck Jones K., Willett P. (Eds.), *Readings in Information Retrieval*, Morgan Kaufmann, San Francisco, 1997, pp. 135–142. Reprinted from RIAO-85 Actes: Recherche d'Informations Assistee par Ordinateur, France IMAG, Grenoble, pp. 645–660.

[27] Dellarocas C., "Building trust on-line: The design of reliable reputation reporting mechanisms for online trading communities", Center for eBusiness@MIT paper 101, 2001.
[28] Denning D., *Information Warfare and Security*, Addison–Wesley, Reading, MA, 1999.
[29] Denning D., "The limits of formal security models", *National Computer Systems Security Award Acceptance Speech*, 1999.
[30] Doob L., *Propaganda, Its Psychology and Technique*, Holt, New York, 1935.
[31] Drineas P., Kerendis I., Raghavan P., *Competitive recommendation systems STOC'02*, May 19–21, 2002.
[32] eBay, Inc. v. Bidder's Edge, Inc., 100 F. Supp. 2d 1058 (ND Cal., 2000).
[33] "Re-engineering in real time", *Economist* (31 January, 2002), http://www.economist.com/surveys/PrinterFriendly.cfm?Story_ID=949093.
[34] Ellul J., *Propaganda*, Knopf, New York, 1966, translated from French by Kellen K., Lerner L.
[35] Farahat A., Nunberg G., Chen F., "AuGEAS (Authoritativeness Grading, Estimation, and Sorting)", in: *Proceedings of the International Conference on Knowledge Management CIKM'02, 4–9 November, McLean, VA*, 2002.
[36] Fawcett T., Provost F., in: Kloesgen W., Zytkow J. (Eds.), *Handbook of Data Mining and Knowledge Discovery*, Oxford Univ. Press, 2002.
[37] Felton E.W., Balfanz D., Dean D., Wallach D., *Web spoofing: An Internet con game. Technical Report 54–96 (revised)*, Department of Computer Science, Princeton University, 1997.
[38] George J., Biros D.P., Burgoon J.K., Nunamaker Jr. J.F., "Training professionals to detect deception", in: *NSF/NIJ Symposium on Intelligence and Security Informatics, June 1–3, 2003, Tucson, AZ*, in: *Lecture, Notes in Computer Science*, Springer-Verlag, Berlin, 2003, pp. 366–370.
[39] Gertz v. Robert Welch, Inc., 428 US 323, 94 S.Ct. 2997, 41 L.Ed.2d 789 (1974).
[40] "Google News beta", http://news.google.com/.
[41] "The Hacktivist. Fluffi Bunni hacker declares Jihad", http://thehacktivist.com/article.php?sid=40, 2001.
[42] Heckman C.J., Wobbrock J., "Put your best face forward: Anthropomorphic agents, e-commerce consumers, and the law", in: *Fourth International Conference on Autonomous Agents, June 3–7, Barcelona, Spain*, 2000.
[43] Herlocker J. (Ed.), *Recommender Systems: Papers and notes from the 2001 workshop, In conjunction with the ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans*, 2001.
[44] Hofmann T., "What people (don't) want", in: *European Conference on Machine Learning (ECML)*, 2001.
[45] Hunt A., *Web Defacement Analysis*, ISTS, 2001.
[46] Huynh D., Karger D., Quan D., "Haystack: A platform for creating, organizing and visualizing information using RDF", in: *Intelligent User Interfaces (IUI)*, 2003.
[47] "Information Warfare Site", http://www.iwar.org.uk/psyops/index.htm, 2001.
[48] *Interpersonal Deception: Theory and Critique, Communication Theory* **6** (3) (1996), special issue.

[49] Johansson P., "User modeling in dialog systems", St. Anna Report SAR 02-2, 2002.
[50] Karlgren J., Cutting D., *Recognizing text genres with simple metrics using discriminant analysis*, 1994.
[51] Kessler B., Nunberg G., Schütze H., "Automatic detection of genre", in: *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, 1997.
[52] Krebs B., "E-mail Scam Sought to defraud PayPal customers", *Newsbytes* (19 December, 2001), http://www.newsbytes.com/news/01/173120.html.
[53] Lafferty J., Chengxiang Z., "Document language models, query models, and risk minimization for information retrieval", in: *2001 ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.
[54] Lafferty J., Chengxiang Z., "Probabilistic relevance models based on document and query generation", in: *Proceedings of the Workshop on Language Modeling and Information Retrieval, Carnegie Mellon University*, 2001, Kluwer volume PT reviewing, in press.
[55] Landwehr C.E., "A security model for military message systems", *ACM Transactions on Computer Systems* **9** (3) (1984).
[56] Landwehr C.E., "Formal models of computer security", *Computing Surveys* **13** (3) (1981).
[57] "Latimes.com., 'Hacktivists', caught in Web of hate, deface Afghan sites", http://www.latimes.com/technology/la-000077258sep27.story?coll=la%2Dheadlines%2Dtechnology, 2001.
[58] Lewis M., "Jonathan Lebed: Stock manipulator, S.E.C. Nemesis—and 15", *New York Times Magazine* (25 February, 2001).
[59] Lewis M., *Next: The Future Just Happened*, Norton, New York, 2001, pp. 35–36.
[60] Libicki M., "The mesh and the Net: Speculations on armed conflict in an age of free silicon", National Defense University, McNair Paper 28, http://www.ndu.edu/ndu/inss/macnair/mcnair28/m028cont.html, 1994.
[61] Lynch C., "When documents deceive: Trust and provenance as new factors for information retrieval in a tangled Web", *Journal of the American Society for Information Science & Technology* **52** (1) (2001) 12–17.
[62] Mann B., "Emulex fraud hurts all", in: *The Motley Fool*, 2000, http://www.fool.com/news/foolplate/2000/foolplate000828.htm.
[63] Maron M.E., Kuhns J.L., "On relevance, probabilistic indexing and information retrieval", *Journal of the ACM* **7** (3) (1960) 216–244.
[64] Mateescu G., Sosonkina M., Thompson P., "A new model for probabilistic information retrieval on the Web", in: *Second SIAM International Conference on Data Mining (SDM 2002). Workshop on Web Analytic*, 2002.
[65] "Matthew Bender and Company, Title 15. Commerce and Trade. Chapter 22. Trademarks general provisions. United States Code Service", http://web.lexis-nexis.com/congcomp/document?_m=46a301efb7693acc36c35058bee8e97d&_docnum=1&wchp=dGLStS-lSlAA&_md5=5929f8114e1a7b40bbe0a7a7ca9d7dea, 2001.

[66] Mensik M., Fresen G., "Vulnerabilities of the Internet: An introduction to the basic legal issues that impact your organization", http://library.lp.findlaw.com/scripts/getfile.pl?file=/firms/bm/bm000007.html, 1996.
[67] Mosteller F., Wallace D.L., *Inference and Disputed Authorship: The Federalist*, Addison–Wesley, Reading, MA, 1964.
[68] MSNBC, "Hacker alters news stories on Yahoo", http://stacks.msnbc.com/news/631231.asp, 2001.
[69] Mundici D., Trombetta A., "Optimal comparison strategies in Ulam's searching game with two errors", *Theoretical Computer Science* **182** (1–2) (1997) (15 August).
[70] Munson J.C., Wimer S., "Watcher: The missing piece of the security puzzle", in: *17th Annual Computer Security Applications Conference (ACSAC'01), December 10–14, New Orleans, LA*, 2001.
[71] "National Center for Digital Government: Integrating Information and Government John F. Kennedy School of Government Harvard University", http://www.ksg.harvard.edu/digitalcenter/.
[72] "National Center for Digital Government: Integrating Information and Government "Identity: The Digital Government Civic Scenario Workshop" Cambridge, MA, April 28–29, 2003, John F. Kennedy School of Government Harvard University", http://www.ksg.harvard.edu/digitalcenter/conference/.
[73] NetworkWorldFusion, "Clever fake of WTO web site harvests e-mail addresses", http://www.nwfusion.com/news/2001/1031wto.htm, 2001.
[74] New York v. Vinolas, 667 N.Y.S.2d 198 (N.Y. Crim. Ct. 1997).
[75] "Newsbytes. Pop singer's death a hoax a top story at CNN", http://www.newsbytes.com/cgi-bin/udt/im.display.printable?client.id=newsbytes&story.id=170973, 2001.
[76] Pratkanis A.R., Aronson E., *Age of Propaganda: The Everyday Use and Abuse of Persuasion*, Freeman, New York, 1992.
[77] Rao J.R., Rhatgi P., "Can pseudonymity really guarantee privacy?", in: *Proceedings of the 9th USENIX Security Symposium, Denver, CO*, August 14–17, 2000.
[78] R.A.V. v. City of St. Paul, 505 U.S. 377, 112 S.Ct. 2538, 120 L.Ed.2d 305, 1992.
[79] "The Register. Intel hacker talks to The Reg", http://www.theregister.co.uk/content/archive/17000.html, 2001.
[80] "The Register. New York Times web site smoked", http://www.theregister.co.uk/content/6/16964.html, 2001.
[81] Rich E., "Users are individuals: Individualizing user models", *International Journal of Man–Machine Studies* **18** (3) (1983) 199–214.
[82] Van Rijsbergen C.J., *Information Retrieval*, second ed., Buttersworth, London, 1979.
[83] Salton G., McGill M., *Introduction to Modern Information Retrieval*, McGraw–Hill, New York, 1983.
[84] Sarwar B., Karypis G., Konstan J., Reidl J., "Item-based collaborative filtering recommendation algorithms", in: *WWW10, Hong Kong*, May 1–5, 2001.
[85] Schneier B., "Semantic attacks: The third wave of network attacks", *Crypto-gram Newsletter* (October 15, 2000), http://www.counterpane.com/crypto-gram-0010.html.
[86] Smith A.K., "Trading in false tips exacts a price", *U.S. News & World Report* (February 5, 2001), p. 40.

[87] Smith S., Personal communication, 2001.
[88] "Sophos. W32/Redesi-B", http://www.sophos.com/virusinfo/analyses/w32redesib.html, 2001.
[89] Thompson P., "Semantic hacking and intelligence and security informatics", in: *NSF/NIJ Symposium on Intelligence and Security Informatics, June 1–3, 2003, Tucson, AZ*, in: *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2003.
[90] Thornton J., "Collaborative filtering research papers", http://jamesthornton.com/cf/, 2001.
[91] Ulam S.M., *Adventures of a Mathematician*, Univ. of California Press, Berkeley, CA, 1991.
[92] Varian H.R., "Resources on collaborative filtering", http://www.sims.berkeley.edu/resources/collab/.
[93] Varian H.R., Resnik P. (Eds.), *CACM* **40** (3) (1997), special issue on recommender systems.
[94] "Washtech.com. FTC shuts down thousands of deceptive web sites", http://www.washtech.com/news/regulation/12829-1.html, 2001.
[95] Wing J.M., "A symbiotic relationship between formal methods and security", in: *Proceedings from Workshops on Computer Security, Fault Tolerance, and Software Assurance*, 1998.
[96] Yahalom R., Klein B., Beth Th., "Trust relationships in secure systems—a distributed authentication perspective", in: *Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland*, 1993.
[97] Yuan Y., Ye E.Z., Smith S., "Web spoofing 2001", Department of Computer Science/Institute for Security Technology Studies, Technical Report TR2001-409, 2001.
[98] Zhou L., Burgoon J.K., Twitchell D.P., "A longitudinal analysis of language behavior of deception in e-mail", in: *NSF/NIJ Symposium on Intelligence and Security Informatics, June 1–3, 2003, Tucson, AZ*, in: *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2003, pp. 102–110.
[99] Zhou L., Twitchell D.P., Qin T., Burgoon J.K., Nunamaker J.F., "An exploratory study into deception in text-based computer-mediated communications", in: *Proceedings of the 36th Hawaii International Conference on Systems Science*, 2003.
[100] "3D Corporation", see http://www.the3dcorp.com/ddd/index2.html.

This page intentionally left blank

The Digital Detective: An Introduction to Digital Forensics[1]

WARREN HARRISON

Portland State University and
Hillsboro Police Department
High Tech Crime Team
Portland, OR 97207-0751
USA
warren@cs.pdx.edu

Abstract

The use of computers to either directly or indirectly store evidence by criminals has become more prevalent as society has become increasingly computerized. It is now routine to find calendars, e-mails, financial account information, detailed plans of crimes, and other artifacts that can be used as evidence in a criminal case stored on a computer's hard drive. Computer forensics is rapidly becoming an essential part of the investigative process, at both local law enforcement levels and federal levels. It is estimated that half of all federal criminal cases require a computer forensics examination. This chapter will address the identification, extraction, and presentation of evidence from electronic media as it is typically performed within law enforcement agencies, describe the current state of the practice, as well as discuss opportunities for new technologies.

[1]The law regarding searching digital devices is complex. This chapter provides an overview of U.S. Federal rules for searching digital devices but is not intended to provide legal advice. The reader is urged to seek competent legal counsel for specific questions.

ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60003-3

1. Introduction

In September 1998, a 53-year old English physician was arrested for the murders of at least 15 of his elderly patients by administering fatal doses of diamorphine, an opiate sometimes used to relieve pain. He would come to be suspected of intentionally killing an additional 200 patients over a 23 year period. The case would turn into the largest serial murder case in UK history.

When investigators searched the offices of the doctor's practice, they found a network of computers running a commercial medical record management system. This system contained the medical records of each of his 3100 patients. Included in each record was a manually entered date and transcription of written notes for every contact the doctor had with each patient. The records of the patients he was suspected of murdering each indicated a lengthy period of declining health, ultimately culminating in their death.

Unbeknownst to the doctor, an upgrade to the medical record management system in October 1996, added an audit trail function. The audit trail recorded every entry made *as well as the date it was entered based on the computer's system clock*. Upon forensic analysis of the computer, investigators found that the record for one patient dated June 23 1997 indicated that she was a chronic morphine abuser. However, upon examining the audit trail file, investigators found that the June 27, 1997 entry was actually entered on June 25, 1998. Not only was the date of the entry falsified, but the patient's body was discovered on June 24, 1998, *the day before the entry was actually entered into the computer.* Similar sorts of entries existed for other patients.

Based partially on the computer evidence as well as other facts and pieces of evidence the physician was convicted. He was sentenced to 15 life terms.

1.1 Computers and Crime

This was not the first case in which incriminating evidence was located on a computer's hard drive, nor will it be the last. Computers have played an increasingly significant role in many crimes committed in the world today. Past-U.S. Attorney General Janet Reno acknowledged the role of computers upon the passage of the U.S. National Information Infrastructure Protection Act of 1996:

> "We see criminals use computers in one of three ways: First, computers are sometimes targeted for theft or destruction of their stored data... Second, computers are used as tools to facilitate traditional offenses... Third, computers are used to store evidence."
> Janet Reno, U.S. Attorney General, Oct 28, 1996

The use of computers to either directly or indirectly store evidence by criminals has become more prevalent as society has become increasingly computerized.

It is now routine to find calendars, e-mails, financial account information, detailed plans of crimes, and other artifacts that can be used as evidence in a criminal case stored on a computer's hard drive. Classical fraud crimes that may have been propagated through the mails or over the telephone in the past are today just as likely to be committed over the Internet. Computer forensics is rapidly becoming an essential part of the investigative process, at both local law enforcement levels and federal levels. In 2002, the FBI expected half of its cases to require at least one computer forensics examination [1].

This chapter will address the identification, extraction, and presentation of evidence from electronic media as it is typically performed within law enforcement agencies, describe the current state of the practice, as well as discuss opportunities for new technologies. The evidence may be of crimes involving computers, such as various forms of identity theft, electronic "stalking" or child pornography, or it may be evidence of a "traditional crime," such as murder or theft, in which the computer is simply a tangential element.

This particular aspect of digital forensics is sometimes known as "media forensics" because it often deals with the extraction of evidence from hard drives or other storage media [2]. There is a great deal of overlap between "media forensics" and "network forensics" where the goal is to identify and investigate network intrusions and cyber attacks. However, many of the details differ. Network forensics often deals with real-time response to network attacks. The goal is usually to determine what was done as a result of the attack, and the appropriate corrective action to take.

This chapter will focus on the extraction of evidence from digital storage devices such as computer disks with generalizations to other devices such as:

- memory cards;
- digital cameras;

- Internet-enabled cell phones;
- PDAs;
- printer or FAX buffers;
- embedded automotive computers.

Usually these examinations would occur well after the fact, and generally there is little of the urgency one might find in recovering from a network attack.

The broad field of issues that may be addressed within digital forensics, as well as the potential for societal good, provides a set of rich opportunities for technologists.

2. Digital Evidence

Digital forensics is concerned with obtaining "relevant evidence" from an electronic medium. "Relevant evidence" is simply any evidence that makes the existence of a fact that is of consequence to the case either more or less probable than it would be without the evidence.[2] This can be as simple as an innocent e-mail between two friends, or as sinister as a set of plans detailing the steps to be carried out to perform a murder.

For instance, assume the defense of a suspect charged with forgery is based on the premise that the forged documents found on his computer were not his but rather belonged to his roommate. The probability of this fact is affected by whether or not the dates and times associated with the last access and/or last modification of those documents occurred at a time when it could be proven the suspect was somewhere other than sitting in front of the computer. If the file access times coincided with dates and times that the suspect was at work, the probability is high that the documents were not his. Therefore, the dates and times of access and modification for a given file may very well be "relevant evidence." This also illustrates a key property of electronic evidence: it has the potential for being both inculpatory (i.e., showing the suspect is guilty) as well as exculpatory (i.e., showing the suspect is innocent).

Obviously a large number of different sorts of electronic artifacts and "meta-artifacts" may serve as relevant evidence. As we have seen, an alibi may be substantiated by time stamped computer logs that put the accused somewhere other than the crime scene when the offense was committed. Likewise, a series of e-mails may indicate a relationship between a victim and a suspect. In fact, just some of the artifacts in which the digital investigator may be interested include [3]:

[2] Strictly speaking an artifact does not become evidence unless its ability to prove a fact has been established. Until then, it is "potential evidence." However, for purposes of convenience, this chapter shall use the term "evidence" to mean "potential evidence" except in cases where the alternate definition is clearly intended.

- e-mail messages;
- chat-room logs;
- ISP records;
- webpages;
- HTTP server logs;
- databases;
- digital directories;
- cookie files;
- word processing documents;
- graphical images;
- spreadsheets;
- address books;
- calendars;
- meta information about files.

Because these artifacts cannot be examined directly with the naked eye (they are, after all, simply electrons recorded on some sort of electromagnetic device) they are what is known as "latent evidence"—that is, evidence that requires equipment, software and/or methods to make it discernable.

2.1 Differences From Traditional Evidentiary Sources

Both the nature and the processing of electronic evidence is dramatically different from other sorts of latent evidence. Electronic evidence is infinitely malleable. While one might destroy other sorts of evidence such as finger prints or the ballistic properties of a bullet, it is highly unlikely that they can be successfully altered. For instance, while a suspect may wear gloves to avoid leaving fingerprints, it is difficult to imagine a way that he could leave someone else's fingerprints. Conversely, it would be equally difficult for an agent of law enforcement to "plant" a suspect's fingerprints at the scene of a crime. However, electronic evidence can easily be altered, and thus it requires particular attention to its *authenticity*.

The process that is used to perform a forensic analysis on a computer is by its very nature different than the nature of the analysis performed by traditional forensic scientists. For instance, a DNA specialist performs basically the same analysis whether the case involves a rape, a murder or a kidnapping. The tools and techniques are the same, as is the science. The DNA specialist can be sure that human DNA will not undergo quarterly upgrades nor that changes in hardware will render DNA obsolete.

Conversely, the digital forensic specialist will perform a very different analysis depending upon the crime. For instance, the approach taken to locate child pornography on a computer is different than that taken to find evidence of identity theft. Techniques and skills will also vary between a Windows desktop computer vs a Linux webserver. Some skills and techniques, such as recovering deleted files from an Apple II will become obsolete in practice after a relatively short time.

2.2 Constraints on Technology

Digital forensics within a criminal justice context is not simply a technical issue. In fact, it is far from simply being a problem with a technical solution. The process used to capture the evidence and the interpretations and conclusions that can be drawn from the results of a forensics examination are subject to strict rules. These rules severely constrain the technological solutions and cannot be ignored. In this chapter we will consider the rules and exclusions in effect in United States Federal Courts [4,5], but the reader can rest assured that similar sorts of issues arise in other jurisdictions and countries.

Should relevant evidence be located in the process of performing a forensic examination of a digital device, the intention is to present that evidence in a trial. However, this is definitely a situation where the "ends do not justify the means." Because the "exclusionary rule" is widely used within the United States legal system, evidence improperly obtained by the government can be excluded from a trial, and once evidence is excluded, it cannot be reintroduced.

Therefore, the process used to obtain, process, and interpret digital evidence is highly relevant to a successful forensics effort, and mistakes in this process can very rarely be undone. These concerns are equally relevant to technologists who want to understand and contribute to the capabilities in this area.

3. The Forensics Process

Reith, Carr, and Gunsch [6] have described a lifecycle model for conducting a digital forensics investigation. The development of such a model is useful both for those involved in a digital forensics effort as well as for providing a taxonomy within which technology and methods can be organized. The Reith, Carr, and Gunsch process model is partially based on the U.S. Federal Bureau of Investigation's Handbook of Forensics Services crime scene search guidelines [7], and extends from the initial recognition of potential evidence through the presentation of this evidence in a trial and its return.

The lifecycle described in this chapter is a slight modification of the one described by Reith, Carr, and Gunsch, in that our discussion ends at the presentation phase since the return of evidence has a more administrative rather than technological flavor. The model described in this chapter also combines the Examination and Analysis phases since from the view of the technologist, both activities appear to be temporally intertwined.

The lifecycle model we will be discussing (see Fig. 1) consists of the following phases:

1. *Identification*. Investigation of any criminal activity may produce digital evidence. This phase deals with the recognition and identification of potential digital evidence.
2. *Preparation*. Once a likely source of digital evidence has been recognized, planning and various preparatory tasks must be carried out. For example, acquiring permission to search is an extremely important piece of preparatory work that must be done before digital evidence can be obtained.
3. *Strategy*. The goal of any effort to collect evidence is to maximize the collection of untainted evidence while minimizing impact of the collection on the victim. In this phase, a plan to acquire the evidence must be developed.
4. *Preservation*. To be useful, the state of physical and digital evidence must be preserved for collection. This phase deals with securing both the physical area as well as the contents of the digital device in question.
5. *Collection*. Ultimately, the digital evidence must be acquired. This is one of the most critical phases of the entire process since it has the most obvious bearing on the authenticity of the evidence.
6. *Examination and Analysis*. Examination involves searching the seized artifacts for possible evidence, while Analysis involves determining the significance of the evidence found, usually within the context of a theory of the crime. The two phases are so intertwined from the technologist's viewpoint that it is difficult to discuss them separately. Examination is affected by the Analysis and vice-versa.
7. *Presentation*. Ultimately, summarization of the conclusions drawn in the Analysis phase as well as explanation of the techniques used in the Collection and Examination activities must be presented to the court. Because jurors cannot be assumed to have any prior technical understanding of computing or computers, this can pose a significant challenge.

In spite of increasingly technical demands, only some of the phases in this process are typically performed by computer specialists. This is especially true for local (i.e., city police and county sheriffs) agencies. Often the digital evidence is seized by personnel from local agencies and then transported to a state or federal agency for

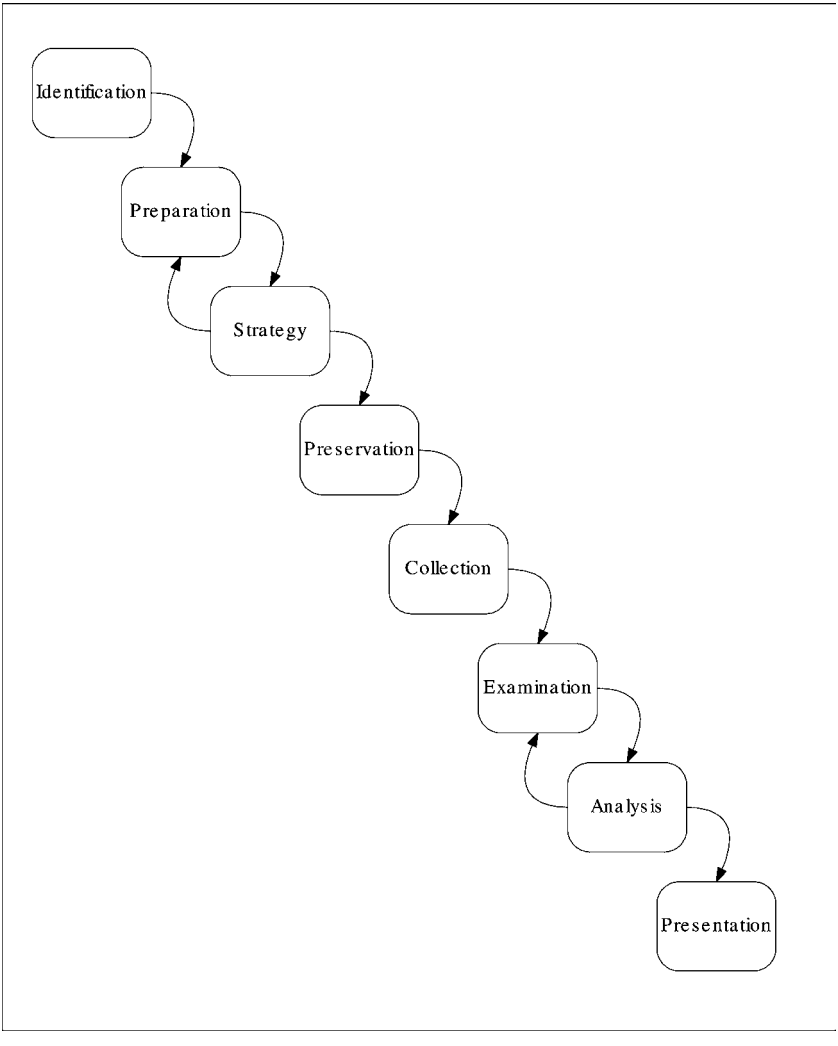


FIG. 1. The digital forensics process.

the Examination phase. In this case, it is likely all the phases through Collection are performed by uniformed officers or detectives with little, if any, special training.

The remainder of this section elaborates upon each of these phases and discusses important issues pertinent to a law enforcement context. However, these issues are

equally relevant to researchers and technical personnel, since they severely constrain the application of both current and future technology.

3.1 The Identification Phase

Investigation of any criminal activity may produce digital evidence. It is important that law enforcement officers recognize, seize, and search digital devices that may contain evidence of a crime.

For example, an officer may notice during a raid on a methamphetamine lab that various state identification cards as well as a computer, color printer and scanner are present. Based on previous experience and training, the officer knows that many methamphetamine addicts support their addiction through identity theft, and that computers and scanners are used to produce false identification. The officer may consider that this computer could potentially contain evidence of the crime of forgery and/or identify theft.

In identifying potential digital evidence, the U.S. Secret Service's "Best Practices for Seizing Electronic Evidence" [8] advises the investigator to determine if:

- *The digital artifacts are contraband or fruits of a crime*. A computer or digital device may contain stolen software or data, or it may contain contraband (i.e., items prohibited by law) such as child pornography.
- *The digital device is an instrumentality of the offense*, or in other words, was the system actively used by the defendant to commit the offense? Computers are often used in identity theft and forgery by scanning stolen identification cards and replacing the actual individual's photo using a product such as Photoshop.
- *The computer system is incidental to the offense*. That is, rather than being used to commit a crime, the computer is used to store evidence of the offense. A murderer may commit meticulous details of a planned crime to a spreadsheet. Drug dealers have been known to maintain digital address books of suppliers and/or customers.

In the methamphetamine lab example described above, the computer will probably be considered to be a potential "instrumentality of the offense" since it is allegedly being using by the suspect to commit the crime of forgery.

Once the digital device's possible role in a crime is understood the investigator can establish if there is *probable cause* to seize the digital device and/or the information it contains. Probable cause is necessary in order to obtain a search warrant, and has several interpretations, depending upon the circumstances.

An interpretation of probable cause that is usually used in connection with search warrants is known as the "*nexus definition*." This interpretation defines probable

cause as some knowledge or observation that would lead a person of reasonable caution to believe that something connected with a crime is on the premises of a person or on person himself. This is augmented by another definition with respect to Law Enforcement in which probable cause is the sum total of layers of information and synthesis of what police have heard, know, or observe as trained officers.

If probable cause exists, then considerations such as if the entire computer, its storage devices, or just the data from the storage devices should be seized. These considerations have important ramifications on the conduct of the subsequent phases in this process and the technology brought to bear in order to extract the evidence. These issues are more fully developed in the next section.

3.2 The Preparation Phase

Once sources of potential digital evidence have been identified, the specifics of how the digital evidence is stored and organized must be determined, For example, is the evidence on a Windows or Linux platform? Is the computer networked? If so, does it entail local or wide area networks? What is the magnitude of the information to be collected? Are the appropriate forensic examination tools or personnel available if the evidence collection must occur on-site rather than at a forensics examination facility?

Even more importantly, the seizure of the evidence must be performed in strict accordance with constitutional and statutory considerations. In the U.S., evidence that is seized inappropriately may be excluded from use in a trial. This is called the "exclusionary rule," and an "airtight case" may evaporate due to lack of evidence if mistakes were made during the search or seizure.

In the United States, the ability of law enforcement agents to search personal property is primarily regulated by the Fourth Amendment to the United States Constitution:[3]

> The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no Warrants shall issue, but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.

[3]While the Fourth Amendment is the primary regulator of searches by government representatives within the United States, other statutory mechanisms also constrain the ability for law enforcement to perform searches. For instance, the Electronic Communications Privacy Act regulates how the government can obtain stored account information from network service providers such as ISPs. The "wiretap statute" and the "the Pen/Trap statute" govern real-time electronic surveillance. A full discussion of these issues is outside the scope of this chapter.

The deciding issue with regards to the Fourth Amendment is if an individual has a *reasonable expectation of privacy*. If a reasonable expectation of privacy exists, then either an individual must provide voluntary consent to the search or a *search warrant* must be obtained to allow a law enforcement officer or their agent to search an item.

A search warrant is a document, issued by a magistrate, giving legal authorization for a search of a container, place or thing. Usually a search warrant narrowly defines the purpose of the search and the items for which the police are searching.

Computers and other digital devices are generally treated like a closed container such as a briefcase or file cabinet [5], and accessing information stored on the device is similar to opening a closed container. In other words, lacking any action on the owner's part minimizing his expectation of privacy this means that law enforcement generally requires a search warrant in order to access the contents of a digital storage device.

However, based on the concept of "plain view" (an officer may search an item that is in "plain view" without consent or a warrant since it is clear that there was no expectation of privacy), certain actions by the owner of the computer may indicate that there was no expectation of privacy. For instance, frequently loaning the computer and user account information to others or taking the computer to a repair shop indicate that the owner did not expect the contents of the computer to remain private.

In general, the courts have ruled that searching files not related to the original warrant (e.g., searching for child pornography when the original warrant was issued to search for evidence of drug sales) exceeds the scope of a warrant.[4] This is significant from a technological point of view because if every file is considered a separate closed container, then all examination of the evidence must be related to the original search warrant.

To illustrate the idea of the scope of a warrant, consider a search warrant that says an officer may search for records substantiating a computer was used by "X," the search of e-mail may be justifiable to determine if "X" ever used the computer to send e-mail. On the other hand, opening graphic and audio files would be much more difficult to justify given the bounds set by this warrant. Likewise, if the computer were to contain the e-mail accounts of a number of other users, there would be no authorization to search the e-mail sent or received by users other than "X."

In order to obtain a search warrant, a sworn statement that explains the basis for the belief that the search is justified by probable cause (the *affidavit*) and the proposed warrant that describes the place to be searched and the things to be seized are submitted to a magistrate. If the magistrate agrees that probable cause has been established and the warrant is not unreasonably broad, they will approve the warrant.

[4]If in the course of carrying out a search, if the examiner legitimately stumbles across evidence of another crime, it will usually serve as probable cause for issuing a new warrant authorizing a search for evidence of that crime.

Because the warrant must not be unreasonably broad (to prevent "fishing expeditions"), special care must be taken when describing digital information and/or the hardware that is to be seized.

If the artifacts to be seized relate to information (e.g., lists of drug suppliers or on-line purchases), it is generally advised that the warrant should describe the information rather than the storage devices on which it resides. Unless the computer in question contains contraband, is an instrumentality or is the fruit of a crime, it is usually advisable to extract the information and leave the computer if at all possible. This is especially true if the computer containing the information is used in a legitimate business or is not owned by the suspect. For instance, it is not at all uncommon for computers owned by the suspect's employer or friend to contain potential evidence. Courts have begun requiring computers seized from third parties and businesses to be either examined in-place or promptly returned.

The information may be specified in very particular terms (e.g., "the cookies.txt file") or may be very broad (e.g., "all evidence of the user visiting a given web site," which may include the cookies.txt file, but also the browser cache, and perhaps even e-mails if use of the site entailed an e-mail confirmation). On the other hand, a request for permission to search "all files for proof of criminal activity" on the computer would likely be construed as too broad to obtain a warrant. Further, even if such a warrant were issued, it is unlikely that evidence collected under the warrant would be considered admissible by the trial court, or that the warrant would withstand an appeal.

If the focus of the search is information and not a particular piece of hardware, specifying information rather than hardware may also allow broader seizure. For example, seizure of information from all computers at a location the suspect may have reasonably used to access the given website may be permitted if information rather than hardware is described. On the other hand, specification of a single desktop computer may require information from the suspect's laptop to be left behind.

If the digital device contains contraband, is an instrumentality as in the methamphetamine lab example or is the fruit of a crime, the device itself will probably be seized and the data extracted off-site. If discovery of the computer and/or peripherals is incidental to a search for a methamphetamine lab and not included in the original warrant, a new warrant would be required to seize these items.

3.3 The Strategy Phase

The strategy phase overlaps at least partially with the preparation phase. In the strategy phase, a decision is made on the approach to be taken to seize the evidence and how it will be carried out.

If the hardware is an instrumentality, contraband, or fruit of a crime, agents will usually plan to seize the hardware and search its contents off-site. This is probably the simplest and most straightforward approach, since it mainly consists of "tagging and bagging" the various devices to be seized.

This strategy also minimizes the level of special skills necessary to carry it out. The U.S. Secret Service publishes a very informative guide for officers and detectives without any special computer knowledge or skills to help them identify and seize computers and other digital devices [8].

On the other hand, if just the information is needed and the hardware is merely a storage device for evidence, it is usually considered wise to extract the information on-site if at all possible. Often times a warrant is served not against a suspect, but rather against a third party such as an employer, a relative or a friend whose computer the suspect used. Extracting the information on-site avoids depriving a (possibly innocent) owner of his computer. It also minimizes disruption of operations if the computer is a business' file server or a mail server for multiple users.

However, collecting the information (as opposed to the computer) on-site is the most complex approach and requires the greatest amount of skill and knowledge by the search and seizure personnel. This approach would probably entail making duplicates of every hard drive and/or storage device in use. Since current storage devices often contain hundreds of gigabytes of data (especially servers), this can be quite time consuming, especially when done on-site. In some cases it may actually be less disruptive to collect the information from the computer off-site.

In the case of the methamphetamine lab, the computer will be considered an instrumentality of a crime. Therefore, the strategy will probably entail labeling and disconnecting the cables from the computer, scanner and printer, boxing, tagging, and physically transporting all three devices, any associated cables and documentation, etc. to the police station or forensics laboratory.

3.4 The Preservation Phase

In order to be useful as evidence, the original state of all digital evidence must be preserved. Because electronic data is so volatile, digital evidence must be promptly secured in order to avoid loss of evidence. A few keystrokes can delete or alter valuable evidence.

Since digital devices are often taken into custody without the benefit of a computer expert, the primary goal is to preserve the evidence for later analysis by a digital forensics expert. The *Secret Service Best Practice Guidelines* instruct the investigator faced with seizing potential electronic evidence involving a computer to:

- *Secure the scene to preserve the state of all digital evidence* by immediately restricting access to all computers to prevent data (including meta data such as

file modification times) from being erased or modified. This includes isolating computers from phone lines and other network connections to prevent data from being accessed remotely.

- *Secure the computer for evidence retrieval*, either by leaving the computer "OFF" if it is not turned on, or if the computer is "ON," photographing the screen and then disconnecting all power sources. This is done by unplugging the computer power cord from the wall *and* the back of the computer.[5] This advice comes from the concern that a "logic bomb" may exist such that if a certain shutdown sequence is not followed, all incriminating files are automatically erased from the computer's hard drive, much like bookies often record their bets on flash paper so in the event of a raid, evidence can be obliterated by simply touching a match to the pages. By simply removing power from the computer logic bombs that may exist in the computer's shutdown scripts can be bypassed.

  Of course, by turning off a running computer, some fragile, transient data may be lost. For example, the content of buffers will be lost, and virtual network drives will be lost. Consequently, some digital forensics experts advise attaching a SCSI device or using an open network connection to get the results of various commands and the contents of various environment variables before turning off a running computer.

In the event the computer being seized is networked or used for business, it is advisable for seizure activities to be assigned to a computer specialist in order to avoid disrupting legitimate business operations while preserving important evidence.

3.5 The Collection Phase

Only after the computer and its immediate surroundings have been secured, can the digital evidence be collected. This may entail taking the computer, taking storage devices connected to it, or in some cases—especially when the computer is used for business purposes—simply taking a copy of the information stored on the computer. In the vast majority of the cases, it is actually the information—not the computer itself or even the storage devices—that is, the evidence in which investigators are interested.

The overriding goal of all the effort expended to preserve the scene and the digital device is to ensure the integrity of the digital content that will be used as evidence. *Digital integrity* can be defined as, "the property whereby digital data has not been

[5]This assumes that a computer specialist is not available on the scene. If a computer expert is available, while their goal will continue to be to preserve all digital evidence, other methods may be employed depending upon the situation.

altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source [9]."

The integrity of the digital content may be at risk both before collection and after it has been collected. In the case of preserving precollection integrity, the investigator is dependent upon effective practices during the preservation phase. In particular, this entails preventing both physical access as well as remote access to the computer or other digital device.

3.5.1 Duplicates vs Copies

Postcollection integrity is assured by creating a duplicate of the original storage device for use in the investigation. This is distinct from acquiring a *copy* of the device. A *duplicate* is "an accurate digital reproduction of all data objects contained on an original physical item" where as a *copy* is "an accurate reproduction of information contained on an original physical item, independent of the original physical item [10]."

To better understand the difference between a duplicate and a copy, it is useful to understand how files are created. In most common computer operating systems a disk consists of unallocated space (i.e., space that currently does not contain the contents of a file) and allocated space (i.e., space which contains data associated with a specific file). If the "unallocated" space has been previously allocated, it probably contains the contents of the file to which the space had most recently been assigned.

In general, files are usually allocated space in fixed sized blocks of memory called clusters. For instance, in the NTFS file system, clusters range from 512 to 4096 bytes, depending on the size of the disk. For most modern disks (over 2 GB in size) the default NTFS cluster size is 4096 bytes. Therefore, the actual amount of space allocated to a file under NTFS will usually be a multiple of 4096 bytes.

Clusters in turn are comprised of sectors, which is the unit of physical transfer that occurs when NTFS performs a read or write operation. A sector is typically 512 bytes in size. Consequently, when a file is created under NTFS, data is written an entire sector (512 bytes) at a time regardless if the amount of data to be written is actually less than 512 bytes.

For example, if an application opens a file for creation, it will initially allocate 4096 bytes (8 sectors) for the file. If the application then writes the string "hello world" to the file and closes it, 512 bytes (one sector) will be written to the file. However, since the 512 bytes written to the file contains only 11 characters ("hello world"), the sector will be padded out over the remaining 501 bytes using random data taken from the computer's RAM. This is called *RAM Slack* and could contain any data that had appeared in memory since the computer was last booted, such as passwords, images, URLs, etc. Likewise, the remaining 7 sectors initially allocated

when the file was created will be "allocated" (i.e., no other file can occupy these sectors), but will contain only what was there when the file was initially created. This is called *File Slack.*

A *duplicate* of a volume entails a bit-by-bit transfer from one device to another. Consequently, it will contain the same data, RAM slack and file slack as the original. Likewise, it will also include the unallocated space from the original. This allows a forensic analysis to be done as though it were being performed on the original.

On the other hand, a *copy* will contain the data, but the copy may very well contain RAM slack and file slack from the computer that did the copying rather than the slack from the original drive. While a copy may be adequate for file backups and ordinary file transfers, since evidence can reside in files, RAM slack, file slack, file meta-data and erased files, a *duplicate* of the original storage device is typically preferred over a *copy.*

Since creating a *duplicate* entails a "bit-by-bit transfer" (actually it is more accurate to say a "sector-by-sector transfer"), tools to create *duplicates* can ignore the specifics of different file systems, since partitions, directory structures, etc. are all copied from the source device to the destination device with no need for interpretation. On the other hand, creating a *copy* (sometimes called a "backup copy") typically implies interpretation of the original storage device's file system since file content are recognized and copied from source to target.

3.5.2 Ensuring Preservation of Evidence During Collection

Obviously it is important to ensure that the duplication procedure does not modify the original storage device in any way. Therefore, hardware and/or software write-blockers must be used when dealing with operating systems and/or tools that modify the source device (for example, by updating logs or other meta-data) whenever it is accessed.

Creating and using a duplicate of the original storage device ensures that the contents of the original device are not inadvertently changed during the Examination phase.[6] In cases where only the information is seized rather than the storage devices or computer itself, the "original" may, in fact, itself be a *duplicate* (or perhaps even a *copy*). Nevertheless, it is treated as an "original," and duplicates for examination are made from it as if it were actually the original storage device.

It is absolutely vital that when a duplicate of an evidence disk is made, whether it is in the field or at the forensic lab, the duplicate is a sector-by-sector copy of the original and can be proven to be so. There are many tools currently used in practice to perform digital duplication. They range from special-purpose disk duplication

[6]Because the investigator may be as interested in the meta-data as in the content itself, virtually any kind of examination prior to imaging will result in some data alteration.

FIG. 2.

hardware devices designed for use in the field such as the Logicube SF-5000 shown in Fig. 2 to software tools such as the UNIX `dd` command used in special controlled environments.

Unfortunately, while the concept of a bit-by-bit duplicate seems intuitively obvious, in practice the definition is not so clear. For instance, virtually all disks have one or more defective sectors when they are shipped from the factory. However, with 99.999999% of the sectors on the disk still usable, there is no reason to discard the entire drive. These blocks are avoided during use by listing them in a map of bad sectors. Since the defective sectors are different on each hard drive, even the best duplication procedure will fail to ensure that every corresponding bit is the same on both the source and target drives. For instance, Table I indicates sectors containing data as D_i and defective sectors that have been mapped as "X."

Nevertheless, these two disks would be considered duplicates because each sector was copied from the source to the target drive. Both the RAM slack and the file slack would be retained, albeit they may be located in different physical sectors on the two disks. Other forensically permissible differences also may occur. For instance, unless exactly the same model hard drives are used, the target drive may be slightly larger than the source drive, have a different number of cylinders, etc.

TABLE I

Source	D1	D2	D3	D4	X	D5	D6	D7	X	X	D8	X	D9
Target	D1	X	D2	D3	D4	D5	X	X	D6	D7	D8	D9	X

One approach that several commonly used by forensic tools such as New Technologies SafeBack [11], Guidance Software's EnCase [12], and Elliot Spencer's iLook [13], which is freely distributed by the Internal Revenue Service to law enforcement agencies, is to store the duplicate of the device as a single image file. Such an image can accurately reflect the various forms of slack and unallocated space.

Using a disk image as opposed to an actual physical disk to hold the duplicate requires that specific tools be used to process the image files as well as preventing the duplicated drive from actually being used (it is considered a bad idea to actually boot off a drive under examination anyway). However, it allows artifacts such as bad block maps to be ignored, and allows things like disk compression on the image so a sparsely populated 80 GB hard drive might have an image that only consumes 10 GB. Evidence from each of these tools have been admitted as evidence in numerous cases. Therefore, the technology behind images vs actual duplicate physical disks has been accepted by the courts.[7]

A recent project [14] at the National Institute of Standards and Technology (NIST) led by Jim Lyle undertook an extensive effort to specify the desirable behavior of forensic disk duplication tools and evaluated the behavior of a number of frequently used tools against this specification. The specification now provides a formal standard against which tools can be evaluated.

One of the biggest motivations for insisting on seizing a physical disk rather than simply creating a duplicate at the scene for analysis is the anticipation of a challenge in court that the "duplicate" somehow differs from the original. To be safe, many investigators prefer to seize the physical disk so it can be presented later as proof that the forensic evidence was not altered.

One approach suggested to address this concern is to create two duplicates of a hard drive in the presence of the owner or some other disinterested third party. One of the drives, the *control drive*, is labeled and sealed. The label is signed by the owner or third party and is stored in a secure location while the other is used as the original "evidence disk." This protects the examiner against a challenge to the authenticity of the working copy, since the owner's copy could be unsealed and compared to the one examined [15].

Regardless of whether an entire computer, the original disk, or a duplicate of the original disk is obtained from the search and seizure activity, the examination is *only* performed on a duplicate of whatever was obtained. The items seized are immediately placed under physical control in order to establish what is known as the "Chain of Custody." The Chain of Custody ensures that the evidence is accounted for at all times; the passage of evidence from one party to the next is documented; as is the

[7] In most cases, the images are only used for examination anyway. The "best evidence" is still the original disk seized from the suspect's computer.

passage of evidence from one location to the next. The location and possession of controlled items should be traceable from the time they are seized until the time they appear in court as evidence.

3.5.3 Using Cryptographic Hashes to Demonstrate Preservation

To further ensurance that the original has not changed after it enters the Chain of Custody, cryptographic one-way hashes of the storage device are used to ensure that its contents continue to exactly match its original contents. Two devices containing exactly the same bit patterns (or the same device whose contents have not been altered being evaluated at two different points in time) should hash to the same value. On the other hand, if the hashes for the devices do not match, then the content of the devices is not the same.

For instance, upon initially seizing a computer, the investigator may choose to compute a hash of the computer's storage device. Later on, a simple rehash of the device will indicate if the contents of the device have changed at all since entering custody. In fact, the use of a one-way cryptographic hash can take the place of the creation of a "control duplicate" as discussed earlier. Rather than signing the sealed control disk, the owner or disinterested third party can attest that the given one-way hash was computed in their presence.

The use of a one-way, sector-by-sector cryptographic hash is also often done as a "sanity check" on the duplicate itself to ensure that the forensic investigation is working with an accurate and reliable duplicate.

Several one-way cryptographic hash algorithms exist. One of the techniques most commonly used within digital forensics is the MD5 [16] hashing algorithm. The MD5 is an algorithm that is used to generate a unique 128-bit fingerprint of a string of any length, up to and including an entire hard drive. With 2^{128} different possible hash values, it is highly unlikely that any two strings would hash to the same value. Another popular method is the SHA-1 hash [17], though MD5 seems to have more popularity with the forensics community.

While proof that hashes agree at two different points in time is important, equally important is that it is computationally infeasible (at least given the resources available to the average law enforcement agency) to engineer a string that generates a particular "target" hash value. This provides assurance that evidence was not placed on the disk *after* the MD5 hash has been computed and the disk contents simply "padded out" to achieve the desired hash value. This can be used to counter claims by the defense that either improper controls over the evidence or malice on the part of the examiner led to evidence "appearing" that was not actually on the original disk.

Should the case lead to prosecution, another duplicate of the original storage device is made available to the Defense under the commonly accepted rules of discovery for their use in evaluating the evidence put forth by the government. In such a situation, the ability to use a digital signature such as an MD5 hash makes it easy to ensure that everyone (both the Prosecution and the Defense) has access to the same information and no tampering of evidence has occurred.

3.5.4 Duplication Speed and Personnel Resources

While there is growing pressure to avoid seizing complete computers or physical devices, as the size of disks in general use increase, the speed of duplicating a target disk becomes significant. Assuming a typical realized transfer rate for consumer IDE hard drives of 20 MB/s, a 120 GB drive will still take almost two hours to duplicate. If two duplicates of each disk must be made on-site (one as the evidence disk and one as a control disk, as discussed earlier) we can expect the process to take over four hours.

Four hours to complete a duplication may be merely an inconvenience in a lab, where one can start the process, walk away and return four hours later to retrieve the duplicate. However, in a seizure situation, the physical location must be secured until the seizure is complete. This requires officers to secure the site and investigators to oversee the duplication. Consequently, a four hour disk duplication could translate into several person days of effort, something most jurisdictions can ill afford.

3.5.5 Preserving Dynamic Data

A running computer stores a wealth of information that is lost as soon as power is disconnected. The prescribed approach to collecting digital evidence, i.e., "interrupt power to the machine," often results in the loss of significant information. There are basically three types of data that is affected by shutting down or removing power from a computer [18]:

- *Transient data*. Information that will be lost at shutdown, such as open network connections, memory resident programs, etc.
- *Fragile data*. Data that is stored on the hard disk, but can easily be altered, such as command histories and log files.
- *Temporarily accessible data*. Data that is stored on the disk, but that can only be accessed at certain times. For instance, once an encrypted file system is accessed using the cryptographic key, all the files are in plain view. However, once the user logs out or the computer shuts down, the file system reverts to its encrypted state, and the contents of the disk are for all intents and purposes lost.

A mechanism to easily capture this sort information prior to shutting a computer down would acquire a great deal of information that is now usually lost.

Kornblum's First Responder's Evidence Disk (FRED) [17] was developed by the Air Force Office of Special Investigations (AFOSI) to allow a minimally trained First Responders to collect transient information such as lists of open network connections, active processes, active DLLs, and open ports from live systems. FRED also collects fragile data such as MD5 hashes of various system files. FRED consists of a single floppy disk containing a batch file and a number of COTS tools so the First Responder can merely insert the floppy into the computer, run the batch file (which stores its reports back onto the floppy disk), and collect a wide range of data that will be gone once the computer is shut down.

3.5.6 Minimizing Intrusion During Collection

Currently, when collecting potential evidence from a computer, the standard method of acquiring the data is by directly accessing the computer's hard disk. This usually entails disassembling the computer and removing the hard drive even if only to create a duplicate.

Such an intrusive procedure increases the possibility of destroying data as well as property. Pins can be bent and cables can be ruined. RAM chips can be fried by static electricity. Having a mechanism by which the contents of a disk can be reliably duplicated without removing any panels or disconnecting any wires would be quite useful. Unfortunately, no universally applicable solution currently exists.

3.5.7 Forensic Challenges of Ubiquitous Computing

As computing becomes ubiquitous, more and more devices will begin to contain digital evidence, and will pose collection problems, especially to minimally trained First Responders. Some of these new and/or unusual devices include:

- Dongles and smart cards often include encryption keys, passwords, and/or access codes.
- Digital phones, answering machines, and caller ID devices can contain information such as the names and/or phone numbers of callers, last number or numbers called, etc.
- Mobile and cellular telephones can contain a wealth of evidentiary information including the phone numbers of the last calls made and received, last several text messages sent or received, and even the last geographical location in which the phone was used [19].
- PDAs usually contain address books, calendars, and notepads [20].

- Memory and flash cards that are used in digital cameras will contain photographs, but they may also be used to store other digital information as well.
- Printers and FAX machines maintain buffers containing the last items printed or scanned as well as user logs.

Each of these lend themselves to different methods of data capture and different evidentiary opportunities. Almost every device is different. For instance, accessing the buffers in a FAX machine is possible, but requires specific knowledge that is not readily known by the typical forensic analyst.

Developing a generic framework by which various new devices can be examined forensically can avoid having to develop new expertise on every case that uses a different device. Some work towards this is being done [21] through attempts to formalize computer input and output systems using a specification language called Hadley. If successful this may provide a generic view of traditional access schemes such as IDE, EIDE, SCSI, etc.

3.5.8 Expertise and Law Enforcement

Quite often potential digital evidence is obtained by the First Responder. Most law enforcement agencies do not have enough trained computer specialists to adequately cover every possible seizure. Nevertheless, except in the case of simply "bagging and tagging" stand alone computer hardware, specially trained technicians should be used for most collection activities, especially duplication of storage devices and management of cryptographic hashes.

If digital evidence *is* obtained by untrained First Responders, the main goal should be to preserve the state of the device so evidence can later be extracted. The most valuable actions that can be taken by a First Responder are [22]:

- Avoid trying to turn on or otherwise access the device.
- Look for and retrieve batteries, power supplies, chargers, etc.
- Record any information shown on the device's display.
- Ask subject, if possible, for PINs or other codes necessary to access the device.
- Prevent the subject from touching the device for any reason.

3.6 The Examination and Analysis Phases

The goal of the examination phase is to locate relevant evidence. Since relevant evidence relates to helping prove or disprove some fact, a trained forensic technician usually carries out the examination under the direction of the investigator who is responsible for determining what facts are pertinent to the case.

For instance, in the case of the computer and peripherals found in the methamphetamine lab example, the investigator will want to locate evidence that may support the charge of producing false identification. In this case, the investigator will ask the examiner to locate files containing forged documents, files containing templates of ID cards and/or files containing photographs of a variety of individuals suitable for pasting into a scanned identification card, subject to the constraints of the search warrant.

3.6.1 The Forensics Environment

The typical digital forensics workbench consists of an Examination Machine (computer) connected to an external drive bay. The Examination Machine runs the examination tools. The external drive bay contains the Evidence Disk during the actual forensic examination. An external drive bay is necessary since a single Examination Machine may be used to process numerous unique cases. Being able to completely remove and replace Evidence Disks without going to the trouble of opening cases and reconnecting cables ensures that evidence from one case is not contaminated by evidence from another. This arrangement can be seen in Fig. 3.

Even though the examination environment should never explicitly write to the evidence disk, some operating systems will write to a hard drive during *any* data access operation. Because of this, the Evidence Disk should be connected to the Examination Machine via either a software or hardware write blocker.

On a typical personal computer, hard drive operations are initiated by issuing a 0X13 interrupt. Software write blockers typically operate by replacing the 0X13 interrupt handler with a new handler that monitors the requested operation. If the operation would cause a change in the associated hard drive, it is blocked, otherwise it is carried out. The National Institute of Standards and Technology has issued a draft specification and test plan for software write blockers [23].

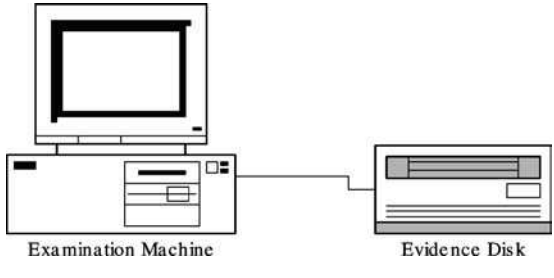


FIG. 3.

Hardware write blockers are simply devices that connect a hard drive to the Examination Machine's bus. The device accepts 'write' commands but fails to act upon them while reporting an acknowledgment that it *has* written the requested data. Without a "success acknowledgment" many Windows applications will hang if they do not receive a signal that the 'write' was completed successfully.

The examination environment may consist of special-purpose forensic GUI-based tools such as EnCase or iLook that provide an interface strikingly similar to a software development IDE (Integrated Development Environment). Consequently, such forensic environments may be referred to as Integrated Forensic Environments (IFEs).

Integrated Forensic Environments can manage all electronic evidence for a case and provide pull-down menus giving access to most of the commonly used forensic functions. Some of these functions include searching for text strings, searching for specific classes of text strings (e.g., e-mail addresses), reconstructing deleted files, matching and excluding "known good" files, etc.

Conversely, the examination environment may simply be a command line interface that allows the examiner to issue commands. Command line environments such as Brian Carrier's @StakeSleuthkit [24] allow extensibility and, as command-line proponents argue, more control over the use of the tools. For example, Fig. 4 shows the use of the @Stake environment on a DOS partition. The mmls command displays the layout of a disk, including the unallocated spaces.

The output of such tools often provide more information to the examiner than an integrated environment. However, they also tend to be cumbersome to use by minimally trained personnel. For example, in order to compute an MD5 hash for a given file, a GUI-based forensic examiner would simply select an option from a pull-down menu while his command-line counterpart would run `md5sum`. As can be expected, both environments have their (vocal) supporters. However, the GUI-

```
% mmls –t dos disk.dd
DOS Partition Table
Units are in 512-byte sectors

     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----   0000000001   0000000062   0000000062   Unallocated
02:  00:00   0000000063   0002056319   0002056257   Win95 FAT32 (0x0B)
03:  00:01   0002056320   0008209214   0006152895   OpenBSD (0xA6)
04:  00:02   0008209215   0019999727   0011790513   FreeBSD (0xA5)
```

FIG. 4. @Stake tools.

based environments have made forensic analysis by non-computer specialists[8] more widespread.

3.6.2 Searching for Evidence

Locating digital evidence typically starts with identifying all computer files or other digital artifacts that have something to do with the case. For instance, in the process of investigating a homicide, a warrant may be executed allowing the contents of a computer disk to be searched for communications between the suspect and the victim.

Pertinent communications may include e-mail, logs of on-line "chat" sessions, letters prepared using a word processor, and perhaps even address books and calendars that establish meetings between the two. If the victim's name is "John Jones," the goal usually is to find all occurrences of the string "Jones" or "John" on the disk.[9] Often this is an iterative process with control moving back and forth between Examination and Analysis.

Often forensic tools can preprocess the evidence disk to create an inverted index in order to speed up searching for single keywords. However, searching for strings in context is still a difficult process. This is one area where the search technology developed for Web-wide Internet searches over unstructured text can be productively brought to bear on the field of digital forensics.

Likewise, even though the examiner knows how to spell *murder* when formulating a search query, there is no guarantee that the suspect does. If the suspect has sent his co-conspirator an e-mail that suggests they *merdur* the victim, the examiner will never discover it by searching for the string *murder*. Application of phonetic encoding techniques such as the Soundex or Double Metaphone [25] algorithms may help identify and locate electronic evidence, regardless of the spelling ability of arbitrary criminals. For instance, Vogon's GenTree [26] forensic environment uses the Soundex algorithm to find strings that "sound like" the string being searched.

The contents of a personal desktop computer may contain thousands of files, and business systems may contain millions. For example, in 2002, the North Texas Regional Computer Forensics Laboratory which services 137 Texas counties processed over 14.6 TB of digital evidence in 474 cases, an average of about 34 cases per examiner and 31 GB of information per case [27].

[8]This is not to imply that users of GUI-based systems have no training—to the contrary, most of the major forensic tool vendors provide training on their tools. However, the GUI framework tends to package the various forensic functions in a manner that is less intimidating to most analysts that come from law enforcement rather than computing.

[9]Given the focus of the search warrant, searches for terms such as "ID Theft" or "Kiddie Porn," would be outside the scope of the warrant and require a new or amended warrant to pursue.

Because of the size of today's digital storage devices, the examiner must use some systematic approach to search for specific textual strings among the thousands of files. This usually involves the use of either GUI-based or command line-based software tools that do efficient string or pattern matching.

3.6.3 Information Abstraction

Ordinarily, information can be viewed as existing on a digital storage device at several levels or layers of abstraction [28]:

- *The Physical Media Layer*: Heads, Cylinders, bytes, etc.
- *The Media Management Layer*: Partitions and partition tables.
- *The File System Layer*: Boot sectors, File Allocation Tables, Directories.
- *The Application Layer*: This usually deals with files stored in some logical content, such as spreadsheet files, word processor documents, address books, and calendar files.

A string search could be applied at any of these levels. For instance, the search could work at the Application, File System or even the Media Management layer. Searches applied at the Media Management Layer can bypass File System conventions such as "deleted files." On the other hand, searches applied at the more abstract level can provide better context for more accurate search results and more easily link the search results to an artifact that can be better understood by nontechnical jurors.

More often than not, multiple searches are performed on an evidence drive. For instance, in the "John Jones Homicide" example one may experiment with several different search strings: "John Jones," "J. Jones," "Jack Jones," "J.J.," etc. When dealing with a few hundred megabytes of files, this is both quick and easy. However, when searching a 120 GB hard drive, each individual search can require hours of processing. Consequently, anything that can reduce the amount of data that must be searched, thereby speeding up the search function, is welcome to both investigators and technicians alike.

3.6.4 Eliminating Known Good

A "known good" file is one that is received directly from the manufacturer or author and has not been modified. Usually these are systems files that are installed when an operating system or other application is installed. Except in rare instances, the contents of these files will have little evidentiary value since a forensic investigation is usually looking for a file that the suspect has either created or modified, such as specific e-mails, spreadsheets, cookies, etc.

TABLE II

Fields	Values for a file from Office 97
File_id	54579
Hashset_id	21
File_name	ORDERS.DBF
Directory	C:\PROGRA~1\MICROS~2\OFFICE
Hash	C5A5113D5493951FE448E8E005A5C136
File_size	989
Date_modified	11/17/96
Time_modified	0:00:00
Time_zone	PST
Date_accessed	12/30/99
Time_accessed	0:00:00

Because we know that known good files will not contain evidence, they can be excluded from string searches. By omitting system and application files from searching, a great deal of time can be saved. For instance, omitting the files installed with Windows XP from a string search can save up to 1.5 GB of searching. Omitting files installed with Office XP can save almost 300 MB.

Identifying known good files can be problematic. If files are excluded based on their file names, it would be an easy job for a criminal to simply rename incriminating files to the names of files installed with popular applications, minimizing the likelihood of a search. The Hashkeeper [29] dataset developed by Brian Deering of the National Drug Intelligence Center introduced the paradigm of using cryptographic hashes, such as the MD5, to uniquely identify files to the forensics community.

The Hashkeeper dataset currently contains MD-5 hashes from hundreds of popular applications that would be expected to be found on most personal user's computer hard drives. These range from Operating System installations such as *Microsoft 2000 Server* to computer games such as *Diablo II* and reference software such as *Broderbund Click Art 10,000*. The dataset accounts for roughly three quarters of a million hashes. A typical entry in the Hashkeeper dataset contains the following comma-delimited fields, see Table II.

An alternate dataset is the National Institute of Standards National Reference Data Library (NRDL) [30]. The NRDL contains MD5 and SHA-1 cryptographic hashes as well as a 32-bit CRC checksum of files from operating systems, vertical applications, database management systems, graphics packages, games, etc. Version 1.4 of the NRDL contains hashes for over 3300 products and over 10,000,000 separate file entries.

The NRDL provides a more sophisticated dataset organization than the Hashkeeper dataset with four comma-delimited tables containing information on Oper-

TABLE III

Fields	Values for a file from WordPerfect
SHA-1	00006DB99FED8A329CC81712584F3949147CCB14
MD5	D48BC5EB79A3FAF08E3A119528F55D72
CRC32	A7A335B4
FileName	hands003.wpg
FileSize	3846
ProductCode	2524
OpSystemCode	WIN

ating Systems, Manufacturers, Products, and of course, the actual file hashes themselves, see Table III.

Prior to beginning any string searches the appropriate hashes of each file on the Evidence Disk can be computed and compared to the hashes in the known good dataset. Files that match the hashes in the known good dataset can be omitted from subsequent string searches.

As long as the hash matching and searches are performed at the File System or Application Layers the idea of "known good" files can be used. However, at less abstract layers, a file is nothing more than a noncontiguous sequence of bits and bytes. Hashes cannot be used under these circumstances to discard files from analysis since there is no way to tell when a file begins or ends.

3.6.5 Searching for Images and Sounds

Searches for arbitrary visual or audio data content present special problems, since there are currently no software tools available to automatically carry out these kinds of searches. For instance, if the examiner in the "John Jones Homicide" example wishes to search for a photo of the suspect with the victim every image file would have to be individually opened and viewed.

To further complicate matters, file names and file-type suffixes are not reliable indicators of the file's contents, so a search for a photo with specific content could potentially entail opening every file on the computer rather than just the ones with obvious graphic file name extensions such as ".gif" and ".jpg."

Again, digital fingerprints in the form of MD5 hashes, come to the rescue. While not every crime has "known bad" files, there are a few that do. In particular, child pornographers are known to continually exchange the same core set of images. Simply possessing pornography containing children is considered illegal since even photographing a child in a sexual context is considered abusive. Because the same set of images regularly appear on these offenders' computers, MD5 hashes have been

developed for a number of well-known images that have been documented to portray minors in pornographic displays.

The use of "known bad" file hashes is particularly significant because child pornography represents a very large percentage of all criminal digital forensic effort—some have estimated as much as 60–70% of the effort expended in examining digital evidence involves child pornography.

In addition to speeding up the search, the collection of hashed images typically used by law enforcement have also been documented to contain minors through identification and interview of the subjects. This is particularly important in the United States where the Supreme Court has ruled the possession of "synthetic child pornography" (the manipulation of nonsexual images of children into images of them engaging in sex acts—for example, by pasting the face of a child onto the body of an adult using an image manipulation program such as Photoshop) is not a crime.[10] Consequently, documentation of images as actually depicting child porn can be very labor intensive. Once an image is documented, using the MD5 hash to uniquely identify it provides an additional degree of efficiency when investigating child pornography.

As is the case in identifying "known good," tools exist that compute and compare the MD5 hash for every file on a computer against a list of MD5 hashes of documented images of known child pornography.

The use of cryptographic hashes to represent specific images also circumvents the problem of maintaining contraband items. By simply maintaining the database of hashes (as opposed to the images themselves) agencies can determine if a suspect possesses child porn on his computer while avoiding the security and control necessary if contraband is maintained on-site.

Unfortunately, even small changes such as cropping can alter the MD5 hash of an image, so it is quite easy to circumvent "known bad" searches. However, hashing "regions" and comparing those for bit mapped images such as JPEGs may help address this problem. However, no current "known bad" hash sets have taken this approach. Further, because most crimes do not lend themselves to a database of "known bad" images this technique has limited applicability.

Another common practice among criminals in the possession of incriminating images (e.g., state identification templates for use in identify theft or forgery) is to change the file name and extension. For instance, the image file "ODL_Template.jpg" can easily be renamed "OT.doc." This is frequently done in the hopes that an examiner may miss the file when manually examining images.

[10]The 1996 Child Pornography Prevention Act originally had made possession of synthetic child pornography illegal, but the U.S. Supreme Court struck this aspect of the law down as a violation of First Amendment rights in April 2002.

Forensic tools are often used that compare file names with the contents of the file. A common image format is the JPEG (Joint Picture Expert's Group File Interchange Format) format. JPEG image files begin with the following 4 byte header:

- FFD8 (Start Of Image marker);
- FFE0 (JFIF marker).

As files are analyzed by forensic search tools, the first four bytes of each file can be compared against the file name extension. Files that present discrepancies, such as a word processing document that actually begins with a JPEG header can be flagged specifically for manual examination. Ironically, the very act of attempting to obfuscate an incriminating file actually draws attention to it during a forensic examination.

Some forensic applications provide galleries of thumbnails for every image on the Evidence Disk as can be seen in the accompanying thumbnail screen (see Fig. 5). The forensic examiner can quickly peruse a screen full of images at a time, looking

FIG. 5.

for possible evidentiary items. However, Constitutional concerns are an issue, since examining a large set of thumbnails may exceed the scope of a warrant.

Nevertheless, searching images for evidence can be very time consuming. Evidence Disks with tens of thousands of different images are not uncommon.

Fortunately, many graphic images are commercial clipart such as the ones that come with Microsoft Office. Therefore, they may also be omitted from analysis through the use of known good filtering. This may reduce the number of image files the examiner must view by hundreds or even thousands.

3.6.6 Recovering Deleted Files

While incriminating evidence is typically found in files to which the examiner has ready access, often evidence is found in files the user has logically deleted, but which have not been physically removed from the hard drive. Since many criminals are technologically naïve, they mistakenly believe that "deleting a file" physically removes its content. Reconstructing deleted files is a common feature of many current forensic tools, and is relatively easy to do.

For instance, NTFS (Windows NT and Windows 2000)[11] uses a Master File Table (MFT) to keep track of files and directories. The MFT contains an entry of a fixed size (usually 1024–4096 bytes, depending on the volume's cluster size) for every file and directory stored on the volume.

Each MFT entry for a file contains a list of *attributes* such as the MAC times (Modified/Accessed/Created), the file name, the file state ("1" for in-use and "0" for deleted) and either the actual file content (if the attributes and content are less than one cluster in size) or a pointer to the content if it is physically stored elsewhere. Each attribute has a header and a value. A *resident* attribute stores both the header and the content in the MFT entry. A *nonresident* attribute stores the header in the MFT entry and the content value is stored in a consecutive group of sectors called a *cluster*.

The MFT also includes an entry for every directory on the volume. While the format is similar to the MFT entry for a file, the "content" of an MFT entry for a folder contains an index of the files that are contained in the folder.

When a file is deleted by the user, the file's index in its directory's MFT entry is deleted and the indexes below it are moved up, overwriting the index. The file's MFT state attribute is changed to "0." At the same time a new MFT file entry is created renaming the file to [drive][index][extension] and added to the Recycle Bin folder.

[11]This is a simplified discussion of the NTFS file system and how file deletions and recoveries are performed. For a more complete discussion the reader is referred to any of the many books on the subject of the NTFS file system.

The date and time of the deletion as well as the original path and naming information is added to a system file in the Recycle Bin folder called INFO.

Neither the deleted file's MFT entry nor its content are physically deleted. If new entries are added to the MFT the deleted MFT entries are overwritten by NTFS before extending the MFT. However, absent the creation of new files or directories, a deleted file and all its content will continue to be available via both the MFT and the Recycle Bin MFT entry. Even if the deleted MFT entry is overwritten, if the data is nonresident, it will probably still be physically retained on the volume.

While the file is still in the Recycle Bin, it is fully accessible using the Explorer, and no special tools or techniques are necessary. Most computer users do not habitually empty their Recycle Bin every day, and deleted files tend to build up.

However, when the user does empty the Recycle Bin, the Recycle Bin indexes are cleared and the MFT entries for both the files stored in the Recycle Bin and the INFO system file are marked as deleted by setting the file state attribute to "0." However, as with a regular file, the MFT entry for the INFO file will not be overwritten until new files are added.

Even if the MFT entries are entirely obliterated, the contents of nonresident attributes (e.g., the contents of a file) may remain accessible at the media management level for an extended period of time after the file is deleted.

3.6.7 Collecting Evidence from Live Systems

Seizure of a personal computer often results in little, if any, *serious* inconvenience to the owner of the computer. In an optimistic scenario, the disk could be duplicated and the computer returned in a matter of days, minimizing the inconvenience experienced by the suspect. During that period, the user may have been prevented from surfing the Web or playing some games, but the relative harm for the average user is minimal.

For a computer used in a normal business, even having the computer off-site for a day can be unworkable. If a company's records are on the computer, business may come to a complete stop until the computer is returned. Small businesses could conceivably face bankruptcy if operations are interrupted for more than a few days. In these circumstances, inconvenience can be minimized by seizing the computer at the end of the business day, duplicating its disks over night and returning the computer the next morning.

However, businesses that provide computing services of various kinds to a large number of users often must have their systems available 24 hours a day, 7 days a week. For instance, a Web hosting company could suffer irreparable damage if the service were shut down for the four to eight hours it may take to make duplicates of its hard drives (actually given the nature of such systems, it would probably take much longer to duplicate their storage volumes).

Additionally, such a system may have hundreds of users, all of whom have a reasonable expectation of the privacy of the information they store in "their" directories. Typically a search warrant would not provide blanket permission to simply snoop through all these unrelated files, and in fact, special rules exist with respect to "private electronic communication" (i.e., e-mail) on "remote computing services" (i.e., service providers such as ISPs).

Significant issues exist regarding how one can acquire admissible evidence from a live, multi-user system without unacceptably degrading performance, while at the same time avoiding access to "unnecessary information." Obviously a live system is continually modifying logs, processes start and stop, files are created, accessed, and deleted. While the technical aspects of capturing such information is relatively easy, it may be difficult to demonstrate authenticity when the case comes to trial.

3.7 The Presentation Phase

Once the Examination and Analysis phases have been completed and the case moves into prosecution, the digital information that has been collected must be admissible to the court in order to have any evidentiary value. The Federal Rules of Evidence [31] establish guidelines for ruling on the admissibility of a specific piece of evidence. Normally, in order to admit the content of a written document, an audio recording, or a photograph into evidence, the *original* written document, audio recording or photograph is required. This is known as the *Best Evidence Rule*.

3.7.1 Best Evidence

Obviously the Best Evidence Rule poses a significant problem when introducing digital evidence into a trial since the evidence itself consist of a collection of electrons that cannot be viewed without some sort of viewing tool. Further, even if one could visually perceive the electrons, they still must be translated using either a text interchange code, such as ASCII or Unicode, or an application that interprets their meaning (e.g., in order for the contents of an ACCESS database to make sense, it must be viewed using the ACCESS database application).

Fortunately, there is an exception to the Best Evidence Rule that allows a printout or other sort of output readable by sight and shown to accurately reflect the data to be considered "Best Evidence" when dealing with data stored on a digital storage device. In other words, an accurate printout of digital information always satisfies the best evidence rule.

3.7.2 Authenticity

Before specific digital information can be admitted as evidence it must be shown to be authentic, or in other words, it must be shown to actually be what it is claimed to be. This is usually done by the testimony of someone that has first-hand knowledge of the digital information. For example, a police officer can testify that a hard drive is the same one that was seized from a computer in the defendant's residence, or a bank officer can testify to the authenticity of bank records.

Challenges to the authenticity (and therefore admissibility) of digital information often take on one of three forms:

1. Digital information can be easily altered, and it can be suggested by the Defense that digital information may have been changed or altered (either maliciously or inadvertently) after the information was seized. However, without specific evidence that tampering occurred such as MD5 hashes that do not match, the mere *possibility* of tampering has been ruled to not affect the authenticity of digital information.
2. If digital information such as logs, meta data, a file's last modified date, etc. has been created by a computer program the authenticity of the information hinges on the reliability of the computer programs that created the data. If the program is shown to have programming errors which could lead to its output being inaccurate the information may not be what it is claimed to be. Because programming errors are so prevalent among commercial software this could raise serious problems when introducing *any* computer-generated evidence. However, the courts have indicated that this challenge can be overcome as long as the information can be considered trustworthy. For instance, the trustworthiness of a computer program can be established by showing that users of the program rely on it's output on a regular basis. Once a level of trustworthiness has been established, challenges to the accuracy of the digital information generated by the computer program affect only the weight (i.e., degree to which the jury considers the evidence) of the evidence, not its admissibility.
3. If the digital information consists of the electronically recorded writings of a person such as e-mail, instant messages, word processing documents, or chat room messages, the statements contained in the digital information must be shown to be truthful and accurate. The authenticity of such evidence is usually challenged by questioning the author's identity—in other words, how do we know the person actually is the one that produced the document? Evidence such as ISP logs or the contents of a user's "Sent" mail folder may end up being used to authenticate such evidence.

Some operations the forensic analyst might perform are fairly straightforward, even to a layperson with minimal computer knowledge. On the other hand, many activities the investigator may perform are not so straightforward.

For instance, retrieving deleted files requires a certain level of technical expertise in order to understand how reliable and valid the technique is. For instance, one might question the likelihood that a "retrieved" file containing the string "I did it," is actually the reconstructed last eight bytes of the string "I agreed to return the money, I did, and I am glad I did it." Or, that there were not actually two files, one that contained "ID" and one that contained "IT?" An examiner must be ready to describe and defend the actions taken to "reconstruct" a deleted file since a juror is more likely to have a "reasonable doubt" if they do not understand the procedures undertaken to collect the evidence.

3.7.3 The Daubert Standard

The 1993 *Daubert vs. Merrell Dow Pharmaceuticals, Inc.* case established the standard for scientific testimony (including testimony involving digital forensics). Prior to 1993, the Frye Standard was used to establish admissibility of scientific evidence based on it's "acceptance" by the scientific community. Under *Daubert*, this standard was raised by requiring additional factors before scientific testimony is allowed:

1. Has scientific theory or technique been empirically tested?
2. Has scientific theory or technique been subjected to peer review?
3. What is known regarding potential error rate (both Type I and Type II errors)?
4. What is the expert's qualifications and stature in the scientific community?
5. Does the technique rely upon the special skills and equipment of one expert, or can it be replicated by other experts elsewhere?
6. Can the technique and its results be explained with sufficient clarity and simplicity so that the court and the jury can understand its meaning?

Currently, the *Daubert* ruling is used in Federal Courts and in the courts of a number of states. In 1999, *Kumho Tire vs. Carmichael*, extended *Daubert* to nonscientific expert testimony. Other *non-Daubert* states use a variety of tests in determining whether expert opinion should be admitted into evidence or not. In general, the *Daubert* standard is both the most restrictive and has the most general applicability. Consequently, new techniques should be evaluated under the six rules listed above.

4. An Illustrative Case Study: Credit Card Fraud

A simple hypothetical case study is provided in this section in order to illustrate some of the points discussed earlier in this chapter. In this case study, an individual is suspected of using a stolen credit card to purchase merchandise from an on-line golfing equipment retailer.

The case begins with an on-line purchase of a set of expensive golf clubs. Order Number ODL79365 was placed at http://www.pdxGolfing.com. for a $1045 set of XLNT golf clubs. The order specified it was to be shipped to the address listed for the credit card using Next Day air delivery by a courier service. Since many merchants will only ship merchandise to the billing address associated with the credit card, a common scheme is to specify Next Day delivery so the perpetrator will know when to expect the merchandise. If the shipment is delivered while the card holder is at work, it is a simple matter to intercept the delivery on the victim's front porch.

The day after the order was placed, the suspect was interrupted rifling through items on the credit card holder's porch by neighbors minutes after a courier delivery. An officer was dispatched to the scene. The responding officer completed a report and issued the suspect a summons for Criminal Mischief III, a relatively minor Class C Misdemeanor. The case was assigned to a detective for follow-up. Originally considered a crime of opportunity, when the cardholder returned home from work and the detective learned the credit card had recently been reported stolen, she immediately suspected the crime was more serious than originally thought. Upon contacting http://www.pdxGolfing.com she verified that the order had been placed using the stolen credit card.

At this point, it is known that the golf clubs had been ordered using the stolen credit card. It is also known that the suspect was found handling the clubs immediately after they were delivered to the credit-card holder's address. However, the credit card itself was not found. The problem for the detective is to tie the suspect to the order. The fact to be proven is whether or not the suspect actually placed the order or not.

During an interview with the suspect, the detective noticed a computer connected to the Internet in the living room. Probable cause was established based on the presence of a computer connected to the Internet, the fraudulent order being placed over the Internet, and the suspect's presence at the cardholder's home minutes after a Next Day delivery was made. Therefore, a Magistrate issued a search warrant allowing all Internet connected computers in the household to be searched for evidence of making this particular purchase. In planning the seizure, the investigator described what the digital forensic expert thought was a Microsoft Windows system.

Uniformed Police Officers and a detective arrived at the house at 9 AM the next morning with the warrant. The Officers immediately escorted each resident of the house into the kitchen while the detective searched each room. The only computer

found was in the living room. When located, the computer was already turned off. Since the forensic technician was not able to accompany them on the seizure, the detective labeled each cable with a piece of tape, and carefully disconnected each from the computer. The computer was placed in the trunk of a patrol car and the suspect was given a receipt for all items taken.

The computer was taken to the Police Computer Forensic Lab, where it was tagged and all pertinent serial numbers written down. Afterwards, the forensics technician removed the seized machine's 40 GB hard drive and signed it out in order to maintain the Chain of Custody. The technician selected a disk of comparable size to serve as the examination disk and verified that the drive had been "scrubbed" to remove any data that may be on the disk.

The technician first computed the MD5 hash for the evidence disk.

The technician placed both the evidence (source) disk and the examination (target) disk in a hardware device called a Logicube SF-5000 [32], a special purpose IDE disk duplication tool. The original hard disk was returned to the Evidence Locker where it was stored with the rest of the computer.

The detective asked the technician to locate evidence that would tie the suspect to the order that was placed at http://www.pdxGolfing.com. Upon some investigation, the technician found that the http://www.pdxGolfing.com retail site uses cookies to persistently maintain the shopping cart while the customer is shopping.

A cookie is an entry on the user's hard drive where Internet applications can store information during a session. Usually the cookie entry includes the name of the domain the user was using when the cookie was recorded, as well as selected pieces of information the Internet application may wish to keep available. A cookie may last for only while the session is active, or it may be saved for a longer period of time so the user can come back later and pick up where they left off.

The examination disk was placed in an external drive bay connected to an examination machine and a quick inventory of the files was made. Noting that the suspect used the Microsoft Internet Explorer Web browser, the forensic technician examined the \Documents and Settings\user\Cookies directory. If the user had used Internet Explorer to place the order, this directory would contain a cookie providing evidence of this fact.

4.1 Alternate Scenario 1

Occasionally, the forensic investigator gets lucky and the suspect does a poor job of covering their tracks. Sometimes it is because the criminal does not think they will ever be caught and other times it is because they are too computer naïve to realize that almost everything a person does on a computer leaves some record behind.

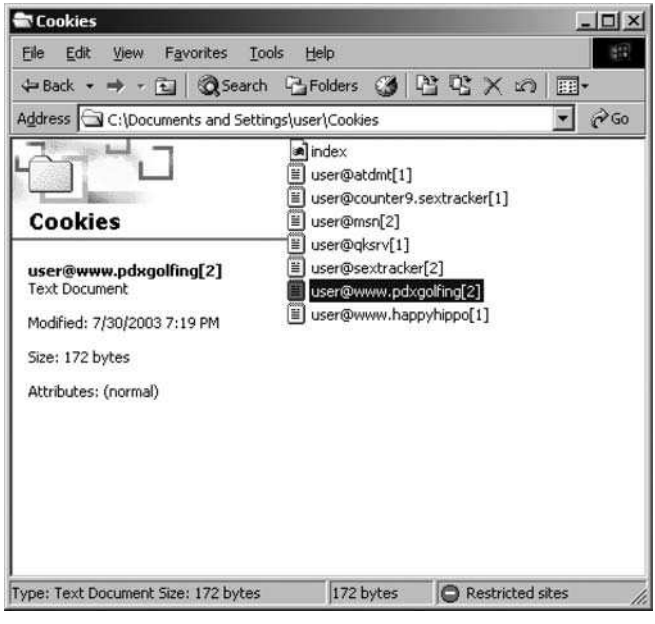


FIG. 6.

As soon as the technician opened the Cookies folder, he found a file called `user@www.pdxgolfing.com[2].txt`, see Fig. 6.

This file contains the cookies deposited by applications located at the pdxgolfing.com domain. Upon opening the file, the technician saw the following two lines:

```
ordernumberODL79365www.pdxgolfing.com/
153624205657602957881514 6
itemsXLNT_Golf_Clubs-1045www.pdxgolfing.com/
1536242056576029578
```

These entries immediately show not only that the suspect's computer had visited this site before, but that the user had placed order number ODL79365 for a set of XLNT golf clubs.

After the forensic analysis, the detective has sufficient evidence to link the suspect with the fraudulent order placed on the Internet. This escalates the crimes the suspect can be charged with from Criminal Mischief III, a Class C Misdemeanor, to at least three Class C Felonies:[12] Computer Crime, Fraudulent Use of a Credit Card

[12]These crimes are based on Oregon Revised Statutes: ORS 164.377, ORS 165.055 and ORS 165.800, the specific crimes the suspect may be charged with will vary from state to state.

(a Class C Felony since the amount is over $750), as well as Identify Theft. This has dramatically raised the stakes for the suspect from a fine of under $1000 and/or 30 days in the county jail to three counts, each of which could result in a fine of up to $100,000 and/or 5 years in prison.

4.2 Alternate Scenario 2

Sometimes a suspect will become suspicious or have foreknowledge of a search. For example, in this case, the suspect already knew that he was under suspicion. This left him ample time to attempt to get rid of incriminating evidence. While he was careful to delete incriminating cookies, he failed to realize the cookies were not actually deleted, but were rather placed into a protected Recycle Bin folder, see Fig. 7.

Much to the suspect's chagrin, the incriminating cookie file can be easily recovered from the Recycle Bin.

However, the Recycle Bin not only preserves the file, but it also preserves the timestamps showing when the file was deleted. If the suspect later claims someone else used his computer to place the order, the deletion time stamp may be able to be

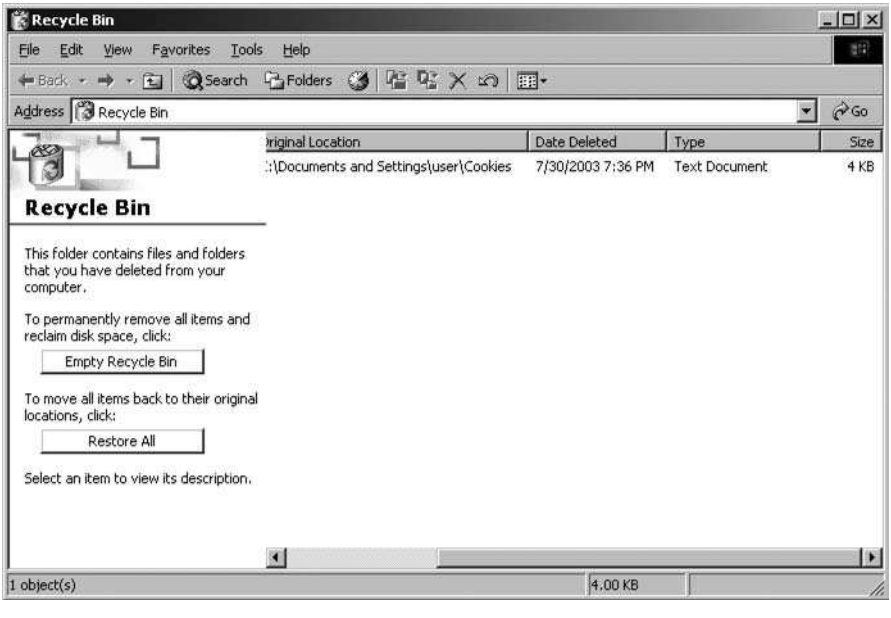


FIG. 7.

used to show that he was the only one with access to the computer at this particular point in time. If this is indeed the case, it would cast serious doubt upon his claim that he had nothing to do with the on-line purchase.

4.3 Alternate Scenario 3

If the suspect is astute enough to think of deleting the cookie files, he is probably astute enough to know that deleted files could be recovered from the Recycle Bin. Therefore, it would be almost certain he would have emptied the Recycle Bin after deleting the cookie file. However, since the suspect's computer used Microsoft Windows 2000, many times even files that are emptied from the Recycle Bin can be recovered.

If a file is small enough, it is stored as an MFT resident attribute rather than being stored in a nonresident cluster. Since the MFT entries on this computer's 40 GB hard drive are 4096 bytes in size, and the actual data stored in the cookie is less than 150 bytes, the examiner can count on the file contents to be part of the MFT entry ($DAT). Therefore, the examiner would likely use one of many available tools to scan for the string `www.pdxgolfing.com` at the media management level, ignoring file boundaries and MFT entries. If the sectors containing the MFT entry have not yet been overwritten, the cookie will almost certainly be found this way.

Once the MFT entry is located, the examiner can trace backwards to find the deletion date. When a file is placed into the Recycle Bin the file name is changed to `[drive][index][extension]`. The MFT entry just found will actually contain a filename of this form (e.g., "`C1txt`") rather than the original file name `user@www.pdxgolfing.com[2].txt`. The Recycle Bin's INFO file lists files and their deletion times, so a second scan for the file name "`C1txt`" will probably turn up the deleted INFO file that contains the file deletion time. Again, as in Alternate Outcome 2, it can be demonstrated that the suspect took steps to remove the evidence of the on-line purchase.

Regardless of which scenario transpires, the forensic examiner still is able to extract the cookie that links the computer, with the on-line order from http://www.pdxgolfing.com. However, if the case goes to trial, the examiner will have to be able to plainly describe the process they used in order to recover the cookie under Alternate Scenario 3. Even though the text found on the hard drive is suspicious, there is still can be a margin of doubt that the text found was in fact deposited as a cookie, unless it can actually be linked to a file named `user@www.pdxgolfing.com[2].txt`.

5. Law Enforcement and Digital Forensics

Most digital forensic investigators come from the ranks of police detectives who have demonstrated an interest in, and aptitude for, using computers. In a National Institute of Justice sponsored report addressing law enforcement's capacity for dealing with electronic crime [33], the top ten issues identified were:

- Insufficient resources to establish computer crime units, pursue investigations and prosecutions, and develop tools.
- Lack of knowledge on how to structure a computer crime unit.
- Lack of uniform training for personnel.
- Inability to retain trained personnel.
- Lack of knowledge of existing technical tools or other resources.
- Insufficient technical tools.
- Lack of standards and certification for technical tools.
- Lack of standards and certification for training.
- Need for updated laws and regulations.
- Insufficient cooperation with the private sector and academia.

It is notable that few of these issues relate to "technology gaps." Many of the issues are far too fundamental. For example, training and awareness of existing resources is necessary before new technologies can be brought to bear.

As the initial wave of forensic specialist pioneers of the 1980s and 1990s retire and the use of digital evidence increases, the make up of the field is changing. For instance, a Federal Bureau of Investigation Forensic Examiner now requires a baccalaureate degree with at least 20 semester hours in computer science, information systems analysis, science/technology, information management, mathematics, computer engineering, or electrical/electronic engineering. An undergraduate Computer Science degree may now be used to qualify for consideration as a Special Agent, the FBI's front-line law enforcement personnel (until 2002, Special Agents generally required degrees in either Accounting or Law).

Interest is also growing by many universities in developing computer forensics curricula at both the graduate and undergraduate level. However, currently there appears to be relatively little interaction between computer science and criminal justice departments.

The characteristics of both current and future practitioners is significant from the perspective of the technologist. The practitioners within a field and their abilities to utilize advanced technology constrain the use and nature of tools every much as do the limitations stemming from the laws on search and seizure.

6. Organizational Structures of Digital Forensics Capabilities

Several organizational models exist for a digital forensics capability within Law Enforcement. There is not currently any one model that has become standardized. The organizational model is important because it heavily influences the level of expertise, training and resources that may be brought to bear in creating and using new technology.

One popular approach is a distributed structure. In this organizational structure, individual agencies each have their own physical facilities (often just a small room, a forensics computer and an external drive) and one or two investigators that have been trained at some level in the technology of digital forensics. In the distributed organizational model, agencies work on their own cases, sometimes sharing resources with other nearby agencies as time and circumstances permit.

Another common approach is a cooperative model in which both resources and personnel are pooled by several agencies to create a regional digital forensics laboratory. Usually the laboratory services a relatively limited geographical area—perhaps adjoining counties for a metropolitan area. Cases are also pooled with no consideration of jurisdiction. For example, if Smallville Police contributes a trained examiner to the regional laboratory, and later they request services, the case may go to any of the examiners, and not necessarily the one contributed by Smallville. This is the model favored by the Federal Bureau of Investigation's Regional Computer Forensic Laboratories (RCFLs). The RCFL model has proven quite effective at mobilizing resources within a geographic area. Currently, eight FBI RCFLs are either in place or under construction: San Diego, California; Dallas, Texas; Kansas City, Missouri; Chicago, Illinois; Buffalo, New York; Newark, New Jersey; Portland, Oregon; and Salt Lake City, Utah.

A very prevalent model is a "service-based" model. In this model, local agencies do not have any digital forensic capabilities themselves. Rather, they may seize an entire computer and send it to a centralized facility, often the state crime lab, or the Federal Bureau of Investigation [34] for analysis.

Each of these different organizational structures has both strong and weak points. For example, in the distributed structure, individual forensic investigators tend to be relatively isolated, and lack colleagues to "bounce ideas off." On the other hand, the agency has full control over its cases, can track their progress, etc. The cooperative model may result in an agency's cases being given lower priority. On the other hand, investigators from different agencies can work in a collegial environment and create a "critical mass" resulting in innovation and growth.

7. Research Issues in Digital Forensics

Digital forensics has been, up until now, a very practitioner-oriented discipline, usually with a law enforcement perspective. This is important, and perhaps even necessary, since digital forensics exists within a very complicated mosaic of factors that constrain the state of the art. These factors include:

- *Externally imposed rules*—what is technically feasible is not necessarily legally possible. For instance, the strict limitations on searching a suspect's possessions imposed by the Fourth Amendment and various statutory rules means that not all technological advances are usable. For instance, it is possible to introduce Trojans, keystroke loggers, etc. into an unprotected or poorly protected computer over the Internet. Evidence could be easily gathered and crimes could be stopped as they occur. However, such technology is generally worthless because evidence collected in this manner would almost be certainly barred from use in court.
- *Available expertise and resources*—currently, most digital forensic examiners have technical training, but few possess the more fundamental knowledge typically expected from an undergraduate Computer Science major. While sophisticated concepts may be usable, they must be presented with an easy-to-use front end to allow use by examiners. Currently a major debate revolves around the use of command line forensic tools such as "DD" and Windows-based tools such as "Encase" and "iLook." Likewise, the resources and facilities most agencies have available to deal with digital forensics are minimal. Few law enforcement agencies have access to supercomputers or satellites (in spite of the image Hollywood presents). Advances that require examiners to have access to advanced knowledge or computing capabilities will be of little practical assistance.
- *Volatility*—because Digital Forensics is (at least currently) practice-based, new advances are often tied closely with specific operating systems or devices. For instance, the details of the process used to recover deleted files varies from MacOS to Windows XP. Because of rapid changes in technology, new challenges and issues are always rising to the surface. Work that was applicable to MS-DOS systems may be totally irrelevant within a PalmOS environment.

Because of its applied nature, useful research in the field will most certainly manifest itself as tools that can be ultimately used in forensic investigations. However, some of the most promising advancements in this area will likely entail adaptation of work in other fields such as database, algorithms, graphics, and software engineering.

8. Conclusions

This chapter has provided an introduction to the field of digital forensics and a brief overview of some of the technology being used. The field has matured greatly since the 1980s when investigators used Microsoft's DEBUG to search for deleted files on MS-DOS machines, and dealt with 5 MB hard drives.

The insular past of the community has restricted participation to practitioners until very recently. With the increased participation of computer science researchers, new techniques and capabilities can be expected.

References

[1] Flynn M.K., "Computer crime scenes", *PC Magazine* (19 February 2002).
[2] Yasinac A., Erbacher R., Marks D., Pollitt M., Sommer P., "Computer forensics education", *IEEE Security and Privacy* (July/August 2003) 15–23.
[3] Hosmer C., "Proving the integrity of digital evidence with time", *International Journal of Digital Evidence* **1** (1) (2002).
[4] *U.S. Federal Rules of Evidence*, U.S. Government Printing Office, 1 December 2001.
[5] "Searching and seizing computers and obtaining electronic evidence in criminal investigations, computer crime and intellectual property section, Criminal Division, United States Department of Justice". Available at: http://www.usdoj.gov/criminal/cybercrime/s&smanual2002.htm, July 2002.
[6] Reith M., Carr C., Gunsch G., "An examination of digital forensic models", *International Journal of Digital Evidence* **1** (3) (2002).
[7] C. Wade (Ed.), *FBI Crime Scene Search*, 1999. Available at: http://www.fbi.gov/hq/lab/handbook/scene1.htm.
[8] "U.S. Secret Service, Best practices for seizing electronic evidence". Available at: http://www.secretservice.gov/electronic_evidence.shtml, 2002.
[9] Vanstone S., van Oorschot P., Menezes A., *Handbook of Applied Cryptography*, CRC Press, 1997.
[10] Scientific Working Group on Digital Evidence and International Organization on Digital Evidence, "Digital Evidence: standards and principles", *Forensic Science Communications* **2** (2) (2000).
[11] http://www.forensics-intl.com/safeback.html.
[12] http://www.guidancesoftware.com/.
[13] http://www.ilook-forensics.org/.
[14] Lyle J., "NIST CFTT: testing disk imaging", *International Journal of Digital Evidence* **1** (4) (2003).
[15] Bates J., "Fundamentals of computer forensics", *International Journal of Forensic Computing* (January/February 1997). Available at: http://www.forensic-computing.com/archives/fundamentals.

[16] Rivest R., *The MD5 Message-Digest Algorithm*, RFC-1321, MIT LCS and RSA Data Security, Inc., April 1992.
[17] FIPS Publication 180-1, Secure Hash Standard, April 1995.
[18] Kornblum J., "Preservation of fragile Digital Evidence by first responders", in: *2nd Annual Digital Forensics Research Workshop*, August 2002.
[19] Willassen S.Y., "Forensics and the GSM mobile telephone system", *International Journal of Digital Evidence* **2** (4) (2003).
[20] Grand J., "pdd: memory imaging and forensic analysis of Palm OS devices", in: *Proceedings of the 14th Annual Computer Security Incident Handling Conference*, June 2002.
[21] Gerber M.B., Leeson J.J., "Shrinking the Ocean: formalizing I/O methods in modern operating systems", *International Journal of Digital Evidence* **1** (2) (2002).
[22] International Organization on Computer Evidence, "Good practices for seizing electronic devices", in: *Notes from the International Organization on Computer Evidence 2000 Conference, Rosny sous Bois, France*, December 2000.
[23] "Hard Drive Software Write Block Tool Specification and Test Plan", Draft Version 3.0, National Institute of Standards and Technology, May 2003.
[24] http://www.sleuthkit.org/sleuthkit/.
[25] Philips L., "The double metaphone search algorithm", *C/C++ Users Journal* (June 2000).
[26] http://www.vogon-computer-evidence.us/gentree_software.htm.
[27] NTRCFL website: http://www.ntrcfl.org/.
[28] Carrier B., "Defining digital forensic examination and analysis tools using abstraction layers", *International Journal of Digital Evidence* **1** (4) (2003).
[29] http://www.hashkeep.org.
[30] National Institute of Standards, "National Software Reference Library (NSRL) Project Web Site". Available at: http://www.nsrl.nist.gov/index.html.
[31] Kerr O.S., *Computer Records and the Federal Rules of Evidence*, Computer Crime and Intellectual Property Section, Criminal Division, United States Department of Justice, March 2001.
[32] http://www.logicube.com/products/hd_duplication/sf5000.asp.
[33] Stambaugh H., Beaupre D.S., Icove D.J., Baker R., Cassaday W., Williams W.P., "Assessment for State and Local Law Enforcement", U.S. Department of Justice Report, NCJ 186276, March 2001.
[34] "FBI Handbook of Forensic Services, Computer Examinations". Available at: http://www.fbi.gov/hq/lab/handbook/examscmp.htm.

This page intentionally left blank

Survivability: Synergizing Security and Reliability

CRISPIN COWAN

Immunix, Inc.
920 SW 3rd Avenue
Portland, OR 97204
USA
crispin@immunix.com

Abstract

In computer science, reliability is the study of how to build systems that continue to provide service despite some degree of random failure of the system. Security is the study of how to build systems that provide privacy, integrity, and continuation of service. Survivability is a relatively new area of study that seeks to combine the benefits of security and reliability techniques to enhance system survivability in the presence of arbitrary failures, including security failures. Despite apparent similarities, the combination of techniques is not trivial. Despite the difficulty, some success has been achieved, surveyed here.

ADVANCES IN COMPUTERS, VOL. 60
ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60004-5

1. Introduction

At first glance, reliability and security would seem to be closely related. *Security* is defined [71] as privacy, integrity, and continuation of service, the latter seeming to encompass a degree of reliability. Conversely, *reliability* is defined as systems that *mask faults* to prevent *failures* of systems to provide their specified services [70].

The combination reliability and security is a natural fit: both seek to improve system availability, both must deal with failures and their consequences. Techniques to ensure software quality such as source code auditing, type safe languages, fault isolation, and fault injection all work well for both security and reliability purposes; improving one tends to improve the other.

However, interpreting security "faults" (vulnerabilities) as failures in the reliability sense has proven to be problematic. *Failure* is defined as "deviation from specification" which is not helpful if the specification itself is wrong. More over, many real systems are implemented in type-unsafe languages (especially C) and so correspondence to a formal specification cannot easily be assured. Thus reliable software does what it is supposed to do, while secure software does what it is supposed to do, and *nothing else* [2]. The surprising "something else" behaviors form the crux of the software vulnerability problem.

So security and reliability cannot be trivially composed to achieve the benefits of both. *Survivability* is the study of how to combine security and reliability techniques to actually achieve the combined benefits of both. The intersection of the two is to be able to survive failures of the security system. The union of the two is to survive arbitrary failures, including the security system, not just random failures.

The rest of this chapter is organized as follows. Section 2 describes the problem of composing security and reliability techniques in greater detail. Section 3 surveys and classifies survivability techniques. Section 4 surveys methods of assessing survivability. Section 5 describes related work surveying survivability. Section 6 presents conclusions.

2. The Problem: Combining Reliability and Security

Reliability defines a *fault* to be when something goes wrong, and a *failure* as when a fault manifests a consequence to a user, such that the system no longer performs its required function. Thus the principle approach to building reliable systems is *fault masking* in which some provisions are made to prevent individual faults from producing failures. *Fault tolerance* techniques notably achieve fault masking by providing *redundant* components, so that when one component fails, another can take up its burden.

A crucial assumption to fault tolerance is that faults are *independent*: that there is no causal relationship between a fault in one component and a fault in another component. This assumption breaks down with respect to *security faults* (vulnerabilities) because replication of components replicates the defects. Attackers seeking to exploit these vulnerabilities can readily compromise all replicas, inducing failure. Thus survivable systems must provide something more than replication to be able to survive the design and implementation faults that are at the core of the security problem.

Reliability also assumes that faults are random, while security cannot make such an assumption. For instance, a system that depends on random memory accesses not hitting the address 0x12345678 can be highly reliable (assuming no data structures are adjacent) but is not secure, because the attacker can aim at an address that is otherwise improbable. In the general case, the attacker can maliciously induce faults that would otherwise be improbable. Thus the traditional reliability techniques of redundancy and improbability do not work against security threats.

The traditional security approach to masking security faults is prevention: either *implement* with such a high degree of rigor that vulnerabilities (exploitable bugs) do not occur, or else *design* in such a way that any implementation faults that do occur cannot manifest into failures. Saltzer and Schroeder canonicalized these prevention techniques in 1975 [71] into the following security principles:

1. *Economy of mechanism*: designs and implementations should be as small and simple as possible, to minimize opportunities for security faults, i.e., avoid bloat.
2. *Fail-safe defaults*: access decisions should default to "deny" unless explicitly specified, to prevent faults due to unanticipated cases.
3. *Complete mediation*: design such that *all* possible means of access to an object are mediated by security mechanisms.
4. *Open design*: the design should not be secret, and in particular, the design should not *depend on secrecy* for its security, i.e., no "security through obscurity."
5. *Separation of privilege*: if human security decisions require more than one human to make them, then faults due to malfeasance are less likely.
6. *Least privilege*: each operation should be performed with the least amount of privilege necessary to do that operation, minimizing potential failures due to faults in that privileged process, i.e., don't do everything as `root` or `administrator`.
7. *Least common mechanism*: minimize the amount of mechanism common across components.
8. *Psychological acceptability*: security mechanisms must be comprehensible and acceptable to users, or they will be ignored and bypassed.

These principles have held up well over time, but some more than others. Least privilege is a spectacular success, while least common mechanism has failed to compete with an alternate approach of enhanced rigor applied to common components that are then liberally shared.

Unfortunately, these techniques also turn out to be too expensive. They are hard to apply correctly, succeeding only rarely. When they do succeed in building highly secure (invulnerable) systems, the result is so restricted and slow that it tends to fail in the market place, having been eclipsed by less secure but more featureful systems.

So in practice, Saltzer and Schroeder's techniques fail most of all for lack of being applied in the first place. Security faults are thus inevitable [26]. Survivability is then the study of how to mask security faults, and do so such that attackers cannot bypass the fault masking. Section 3 examines how security faults can be effectively masked.

3. Survivability Techniques

In Section 2 we saw that redundancy and improbability are insufficient to mask security faults against an intelligent adversary, because the adversary can deliberately invoke *common mode failures*. How then to mask unknown security faults?

Colonel John R. Boyd, USAF, defined a strategy called *OODA*: Observe, Orient, Decide, and Act [42]. These four steps describe specifically how a fighter pilot should respond to a threat, and the approach generalizes to other fields of conflict, including computer security, which plays out as follows:

1. *Observe* data that might indicate a threat.
2. *Orient* by synthesizing the data into a plausible threat.
3. *Decide* how to react to that threat.
4. *Act* on the decision and fend off the threat.

This strategy is often used in computer survivability research to build *adaptive intrusion response* systems. These systems detect intrusions (using IDS/Intrusion Detection Systems) and take some form of dynamic action to mitigate the intrusion. They precisely follow Boyd's OODA loop.

Survivability techniques vary in the *time frame* in which the intrusion detection and response occur. Boyd advocated *tight* OODA loops to "get inside" the adversary's control loop, acting before the adversary can respond. In computer systems, tight response loops have the distinct advantage of preventing the intrusion from proceeding very far, and thus prevent most of the consequent damage.

However, larger OODA loops are not without merit. Taking a broader view of intrusion events enables more synthesis of what the attacker is trying to do, producing better "orientation" (in Boyd's terms) and thus presumably better "decisions."

In the following sections, we examine survivability techniques in terms of when they mitigate potential security faults. Section 3.1 looks at design time mitigation. Section 3.2 looks at implementation time techniques. Section 3.3 looks at run time intrusion prevention. Section 3.4 looks at intrusion recovery techniques.

3.1 Design Time: Fault Isolation

Despite the problems described in Section 2, security and reliability techniques do have overlap, especially in the area of design: to minimize the scope of faults. The Principle of Least Privilege can be applied in many contexts to enhance both security and reliability, such as:

- address space protection, which isolate wild pointer process faults to that process
- microkernels, which further compartmentalize facilities such as file systems into separate processes, rather than placing them in a single monolithic kernel
- mandatory access controls, which limit the scope of damage if an attacker gains control of a process

Mandatory access controls were the major security result of the 1970s and 1980s. Firewalls were the major security result of the late 1980s and early 1990s, in effect providing mandatory access controls for entire sites at the network layer. This is a natural consequence of the shift to decentralized computing: mandatory access controls are effective when all of the users are logged in to a single large time-share system. Conversely, firewalls are effective if all of the users have their own computer on the network, and thus access must be controlled at the network level. Section 3.1.1 elaborates on mandatory access controls, and Section 3.1.2 elaborates on firewalls.

3.1.1 Mandatory Access Controls

Access controls have always been a fundamental part of secure operating systems, but the devil is in the details. One can achieve true *least privilege* [71] operation by precisely specifying permissions for *every* operation using Lampson's access control matrix [54], in which columns are users (or any active entity) rows are files (or any entity which may be acted upon) and the cells contain the operations that the user may perform on the file. Table I shows a simple example of a Lampson access control matrix, but it is far from complete. Since real systems have many users and hundreds of thousands of files, a complete application of an access control matrix would lead to an unmanageably complex access control matrix, leading to errors in *authorization*.

Thus the goal in access control is to create an *abstraction* for access control that closely matches the anticipated usage patterns of the system, so that access control

specifications can be created that are both *succinct* (easy to create and to verify) and *precise* (closely approximate least privilege).

Access control schemes originally used an abstraction of controlling interactions among *users* on a time-share system. But by the mid-1990s, most computers had become single-user: either single-user workstations, or no-user network servers that do not let users log in at all and instead just offer services such as file service (NAS), web service, DNS, etc., and thus user-based access controls schemes became cumbersome. To that end, survivability research has produced several new access control mechanisms with new abstractions to fit new usage patterns:

- *Type enforcement and DTE*: Type enforcement introduced the idea of abstracting users into domains, abstracting files into types, and managing access control in terms of which domains can access which types [11]. DTE (Domain and Type Enforcement [3,4]) refined this concept.
- *Generic Wrappers*: This extension of DTE [38] allows small access control policies, written in a dialect of C++, to be dynamically inserted into a running kernel. A variation of this concept [5] makes this facility available for Microsoft Windows systems, but in doing so implements the access controls in the DLLs (Dynamically Linked Libraries) instead of in the kernel, compromising the non-bypassability of the mechanism.
- *SubDomain*: SubDomain is access control streamlined for server appliances [23]. It ensures that a server appliance does what it is supposed to and nothing else by enforcing rules that specify which files each program may read from, write to, and execute. In contrast to systems such as DTE and SELinux, SubDomain trades expressiveness for simplicity. SELinux can express more sophisticated policies than SubDomain, and should be used to solve complex multi-user access control problems. On the other hand, SubDomain is easy to manage and readily applicable. For instance, Immunix Inc. entered an Immunix server (including SubDomain) in the Defcon Capture-the-Flag contest [20] in which

TABLE I
LAMPSON'S ACCESS CONTROL MATRIX

	Alice (sysadmin)	Bob (accounting)	Carol (engineering)
`/var/spool/mail/alice`	RW		
`/usr/bin/login`	RWX	X	X
`/etc/motd`	RW	R	R
`/local/personnel/payroll`		RW	
`/local/eng/secret-plans`			RW

SubDomain profiles were wrapped around a broad variety of badly vulnerable software in a period of 10 hours. The resulting system was never penetrated.

An unusual new class of access controls emerging from survivability research is *randomization*, in which some aspect of a system is deliberately randomized to make it difficult for the attacker to read or manipulate. The secret "key" of the current randomization state is then distributed only to known "good" parties [25]. To be effective, the system aspect being randomized must be something that the attacker depends on, and must also be something that can be randomized without breaking the desired functionality of the system. System aspects that have been randomized are:

- *Address space layout*: Forrest et al. [36] introduced address space randomization with random amounts of padding on the stack, with limited effectiveness. The PaX project [82] introduced the ASLR (Address Space Layout Randomization) feature, which randomized the location of several key objects that are often used by buffer overflow and `printf` format string exploits. Bhatkar et al. [10] extended ASLR by randomizing more memory objects, but with a limited prototype implementation. Xu et al. [95] extended in the other direction, providing a degree of address space randomization using *only* a modified program loader.
- *Pointers*: PointGuard [22] provides the exact dual of address space randomization by encrypting pointer values in memory, decrypting pointer values only when they are loaded into registers for dereferencing. Memory layout is left unchanged, but pointers (which act as the effective *lens* through which memory is viewed) are hashed so that only *bona fide* code generated by the compiler can effectively dereference a pointer.
- *Instruction Sets*: Kc et al. [48] provide a mechanism for encrypted instruction sets. In one instance, they encrypt the representation of CPU instructions to be locally unique, so that foreign binary instructions are not feasible to inject. In another instance, they apply similar encryption to PERL keywords, so that script commands cannot be injected over, e.g., web interfaces. The former is largely limited to hypothetical CPUs (other than Transmeta, who make CPUs with programmable instruction sets) while the latter is feasible as a normal feature of conventional software systems.
- *IP Space*: Kewley et al. [50] built a system that randomizes the IP addresses within a LAN. Defenders know the secret pattern of changing IP addresses from time to time, while attackers have to guess. Defenders reported this technique as effective, while attackers reported that it was confounding when they did not know it was being used, but once discovered, the limited address space of IPs (256 for a small LAN) made the guessing attack relatively easy.

Randomization defenses are effectively cryptographic session keys, applied to implicit interfaces. Some form of defense is called for when the attacker has access to one side of the interface, i.e., the attacker supplies data to that interface, making the software processing that input subject to attack. Randomization is an effective defense where the defense is *implicit* and thus classical encrypted communication would be difficult to employ, e.g., within an address space.

Randomization is less effective when the interface in question is *explicit* because classical encryption may well be more effective. The singular advantage that network interface randomization offers over network encryption (Virtual Private Networks) is resistance to DoS (Denial of Service) attacks: attackers can flood encrypted network ports with traffic without having any encryption keys, but it is relatively difficult to flood an unknown IP address.

3.1.2 Firewalls

Initially designed to protect LANs from the outside Internet [19], firewalls progressed to being deployed within LANs to compartmentalize them, much the way mandatory access control systems compartmentalized time-share users [8]. In recent self-identified survivability research, the Secure Computing ADF card [67,65] isolates security breaches inside insecure PCs by providing an enforcing firewall NIC managed from somewhere other than the PC itself. Thus compromised machines can be contained from a management console.

In early 2000, a new class of attacks appeared: DDoS (Distributed Denial-of-Service) attacks, in which an attacker co-opts a large number of weakly defended computers around the Internet, and then commands them all to flood a victim's machine with traffic. These attacks are very difficult to defend against, especially for public web sites intended to allow anyone on the Internet to submit requests. Defenses against such attacks are either to trace back the origin of the attacks [72] or to attempt to filter the DDoS traffic from legitimate traffic somewhere in the network to block the flood [34,66,84]. Work in this area has lead to commercial ventures such as Arbor Networks (www.arbornetworks.com) and Mazu Networks (www.mazunetworks.com).

The problem with DDoS defenses is that they are subject to an arms race. The detection technologies are relying upon synthetic artifacts of the bogus data that the DDoS agents are generating. As the DDoS agents become more sophisticated, the data will come to more closely resemble legitimate traffic. In principle, there is no reason why DDoS traffic cannot be made to look exactly like a heavy load of real traffic: this is a fundamental difference between DoS attacks and misuse attacks, that DoS attacks need not deviate *at all* from real traffic. When faced with DDoS traffic that is identical to real traffic, filtering will become either ineffective or arbitrary.

Traceback has a better chance of success in the face of sophisticated attack, but will be labor-intensive until the Internet itself changes to support effective traceback of traffic.

3.2 Implementation Time: Writing Correct Code

The least damaging vulnerability is one that never ships at all. To that end, many tools have been developed to help find and eliminate software vulnerabilities during the development cycle. Three classes of tools have emerged:

- *Syntactic checkers*: These tools scan the program source code looking for syntactic patterns associated with vulnerabilities, e.g., calls to unsafe functions like `gets()` [87,92].
- *Semantic checkers*: These tools do deeper analysis of program semantics looking for vulnerabilities such as buffer overflows and TOCTTOU (Time of Check to Time of Use) vulnerabilities [89,18,75].
- *Safer Language Dialects*: These are variants of the C programming language intended to make it safer to write programs by making it more difficult to write vulnerable code, e.g., by removing dangerous features like pointer arithmetic [46,64]. Safer C dialects notably do *not* include C++ [80] which adds many convenience features of object oriented programming, but does not have a safe type checking system because it does not remove pointer arithmetic from the language.
- *Safer Languages*: Programming languages such as Java [41] and ML [63] that were designed from the outset to be type safe offer even greater safety and efficiency. By designing the language to facilitate strong type checking, deeper semantic analysis of the program is possible, allowing compilers such as Hermes [78] to even discover inconsistencies in the *state* of some values, precluding some potential run-time failures [79].

3.3 Run Time: Intrusion Prevention

Regardless of fault containment designs, some code must be trusted, and regardless of the implementation techniques used, some vulnerabilities (bugs) will slip through, and so vulnerabilities are inevitable. Thus something must be done to detect the exploitation of these vulnerabilities and (hopefully) halt it. The reliability community calls this *fail-stop* behavior (assuming that the failure is detected). The survivability research community calls it *intrusion detection* and *adaptive response*. The security commercial sector, which has recently become interested, calls it *intrusion prevention*.

There are several dimensions in which intrusion prevention techniques can be classified. We present a 3-dimensional view, intended to classify together technologies that achieve similar goals:

- *Network vs. Host*: Intrusion prevention can be done either at the network layer or within the host. Network intrusion detection is much easier to deploy, but because there is little context in network traffic, it is possible for attackers to evade network intrusion detection methods [69,76].
- *Detection vs. Prevention*: Some tools only detect intrusions, while others respond to intrusion events and shut the attackers down (closing the OODA loop). Prevention is effectively detection + response.
- *Misuse vs. Anomaly*: Some systems characterize and arrest known system misuse and allow everything else (*misuse detection*) while others characterize normal system behavior and characterize *anomalous* behavior as an intrusion. Misuse detection is fast and accurate, but fails to detect *novel* attacks. Anomaly detecting can detect novel attacks, but is subject to a high *false positive* rate (complaints about traffic that is actually legitimate) [57].

Populating this array of properties, we get:

- *Network*
  - *Detection*
    - *Misuse*: This area is dominated by commercial NIDS (Network Intrusion Detection) products such as the commercial ISS RealSecure and the open source SNORT.
    - *Anomaly*: The ability to detect novel attacks has generated keen interest in this area of research [55,86], but little of it has had real-world impact due to high false positive rates. Industrial applications of intrusion detection demand *very* low false-positive rates, because non-trivial false-positive rates combined with high bandwidth lead to high staffing requirements.
  - *Prevention*
    - *Misuse*: Network misuse prevention emerged as a commercial market in 2002, taking the more reliable parts of network intrusion detection systems and placing them *in-line* with the network connection, acting much like an application-level firewall with misuse prevention rules. Example systems include the Hogwash and Inline-SNORT NIPS (Network Intrusion Prevention Systems).
    - *Anomaly*: Network anomaly prevention is a new way of looking at classic firewalls, which permit traffic with specified source and destination IP addresses, ports, and protocols, and deny all other traffic.

- *Host*
  - *Detection*
    - *Misuse*: This area is referred to as HIDS (Host Intrusion Detection System) typified research projects such as EMERALD [68], STAT [88]. These systems actually use a combination of misuse and anomaly detection. Commercial HIDS similarly use a combination of anomaly and misuse detection, and also provide for both detection and prevention, as exemplified by products such as Zone Alarm and Norton Personal Firewall.
    - *Anomaly*: There are a variety of ways to do host anomaly detection, depending on which factors are measured, and how "normal" and "anomalous" are characterized. Forrest et al. [35] monitor sequences of system calls, ignore the arguments to system calls, and look for characteristic n-grams (sequences of length n) to distinguish between "self" (normal) and "non-self" (anomalous). Eskin et al. [32] generalize this technique to look at dynamic window sizes, varying n. Ghosh et al. [40] use machine learning to characterize anomalous program behavior, looking at BSM log records instead of system call patterns. Michael [62] presents an algorithm to find the vocabulary of Program Behavior Data for Anomaly Detection, so as to substantially reduce the volume of raw, redundant anomaly data to be considered. Tripwire [51,45] does not look at program behavior at all, and instead detects changes in files that are not expected to change, based on a checksum; files are profiled in terms of their probability of change, so that, e.g., changes in `/var/spool/mail/jsmith` are ignored (e-mail has arrived) while changes in `/bin/login` are considered very significant.
  - *Prevention*
    - *Misuse*: The most familiar form of host misuse prevention is antivirus software that scans newly arrived data for specific signatures of known malicious software. However, this very narrow form of misuse is also very limited, in that it must be constantly updated with new virus signatures, a limitation made obvious every time a virus becomes widespread before the corresponding signature does, causing a viral bloom of Internet mail such as Melissa [14], "I Love You" [15], or Sobig.F [17]. Host misuse prevention can be either provided by the environment, or compiled in to software components.
      - *Kernel*: In the kernel environment, an exemplary system is the Openwall Linux kernel patch [29] which provides both a non-executable stack segment to resist buffer overflow attacks, and also prevents two pathological misuses of hard links and symbolic links. PaX [82] generalizes Openwall's non-executable stack segment to provide non-executable heap

pages by manipulating the TLB, which is otherwise problematic on x86 CPUs. RaceGuard [24] detects violations of the atomicity of temporary file creation to fend off temporary file race attacks.

+ *Library*: At the library level, Libsafe [6] provides a version of `glibc` that does plausibility checks on the arguments to string manipulation functions to ensure that buffer overflows and format string attacks [83] do not corrupt the caller's activation records.
+ *Compiled In*: Compiled into programs, StackGuard [27] and derived work [33,12] compile C programs to produce executables that detect attempts at "stack smashing" buffer overflow attacks that try to corrupt activation records to hijack the victim program, and fail-stop the program before the attack can take effect. PointGuard [22] generalizes this concept to provide broader coverage by encrypting pointers in memory and decrypting pointer values only when they are loaded into registers for dereferencing. FormatGuard [21] provides a similar protection against `printf` format string vulnerabilities.

∗ *Anomaly*: Operating systems can profile user or application behavior, and arrest anomalous deviations. Similar to network anomaly prevention, this induces sufficient false positives that it is not often deployed. For instance, to address the limitations of antivirus filters described above in Host Misuse Prevention, various survivability research projects [5,43,73] have built systems to limit the behavior of workstation applications that process network input (e.g., mail clients and web browsers) to minimize potential damage if the application is hijacked. Mandatory access controls (MAC) also fit in here (see Section 3.1.1) by using the MAC system to describe what applications or users may do, and then prohibit everything else.

Many of the above technologies require modification to the operating system kernel to be effective. Because of that need, the open source license and wide popularity of the Linux kernel make it a popular target for survivability research. Unfortunately, the result of a research project that customizes the kernel to include survivability features is a "private fork" which is easy enough for researchers to build for their own use, but not sufficiently convenient for IT workers to deploy.

To address this need, the Linux Security Modules project (LSM [94,93]) was built to provide a sufficiently rich loadable kernel module interface that effective access control modules could be built without needing to modify the standard Linux kernel. LSM has now been accepted as a standard feature, first appearing for production in Linux 2.6. As a result, IT workers should be able to obtain survivability-enhancing modules, either open source such as SELinux [37,60] or proprietary such as Sub-

Domain [23] and load them into the standard kernels they get with commercially distributed Linux systems.

Finally, we return to Boyd's OODA Loop. In the above technologies, those marked as "prevention" provide their own built-in intrusion mitigation mechanisms (usually fail-stop) and thus provide a very tight OODA loop. Those marked as "detection" need to be composed with some other form of intrusion mitigation to actually be able to enhance survivability.

This longer OODA loop sacrifices the responsiveness that Boyd so highly prized, in favor of more sophisticated analysis of intrusion events, so as to gain greater precision in discerning actual intrusions from false-alarms due to subtle or ambiguous intrusion event data. For instance, IDIP [74] provides infrastructure and protocols for intrusion sensors (network and host IDS) to communicate with analyzers and mitigators (firewalls embedded throughout the network) to isolate intrusions.

The CIDF (Common Intrusion Detection Framework) project was a consortium-effort of DARPA-funded intrusion detection teams to build a common network language for announcing and processing intrusion events. CIDF tried to provide for generality using Lisp-based S-expressions to express intrusion events. Unfortunately, CIDF was not adopted by intrusion detection vendors outside the DARPA research community. Subsequent attempts to build IETF [9] standards for conveying intrusion events have yet to achieve significant impact.

3.4 Recovery Time: Intrusion Tolerance

An explicit goal of survivability research is to build systems that are *intrusion tolerant*: that an intruder may partially succeed in compromising privacy, integrity, or continuation of service, and yet the system tolerates this intrusion, carrying on providing critical services at a possibly reduced rate, rather than catastrophically failing all at once.

The classic reliability approach to *fault* tolerance is redundancy, so that when one replica fails, another can take up its load until repairs are made. However, as discussed in Section 2, the problem here is that while hardware faults are *independent*, the *vulnerabilities* that lead to security faults (intrusions) are largely *not* independent. Thus the attacker can sweep through an array of redundant services, quickly compromising all replicas.

Therefore, to provide intrusion tolerance, something must be done to prevent the attacker from quickly compromising all replicas, by ensuring that they do not all share the same vulnerabilities. N-version programming, where independent teams are given identical specifications to implement, is a classic fault tolerance technique, and would seem to provide for sufficiently diverse implementation to prevent com-

mon vulnerabilities. Unfortunately, N-version programming suffers from several severe limitations:

- It is *very* expensive, multiplying software development costs nearly by N.
- Theoretically independent software teams unfortunately tend to make similar (and wrong) assumptions, leading to common bugs [52]. Independent implementations of TCP/IP stacks and web browsers have often been shown to have common security vulnerabilities. For instance, nearly all TCP/IP implementations crashed when presented with a datagram larger than 64 KB in size (the "Ping of Death") because most developers read the IP specification, which clearly states that all packets have a maximum size of 64 KB, and few thought to check for over-sized packets.

A related approach is to exploit *natural diversity* by deploying a redundant array comprised of, e.g., a Windows machine, a Linux machine, and a FreeBSD machine. Natural diversity has the advantages of lower cost (because it leverages existing diversity instead of paying for additional redundant effort) and sharing fewer common design and implementation vulnerabilities, because the independent teams did not share a specification. Conversely, natural diversity has no design assurance that diversity has been *delivered* in key areas, e.g., many naturally diverse systems abruptly discovered that they depended on the same version of `zlib` (a data compression library) when it was discovered to be vulnerable [16].

Several survivability projects [58,47,65,96] employ the natural diversity defense. They share a common architecture, in which services are replicated across *heterogeneous* servers, e.g., one Linux server, one Windows server, and one BSD server. A router or load balancer of some kind sits in front of the replicas, routing service requests to one or more of the replicas. The hope is that while one flavor of server might be vulnerable, that the vulnerability will not be common to all of the replicas. The system may use some form of intrusion detection to detect infected servers and remove them from the replica pool. The system may use some form of consensus voting to determine correct results, and to remove minority-view servers from the pool as infected.

Often omitted from such systems is a mechanism to share dynamic state across the replicas in a consistent and coherent manner, so that they can provide active content. This is an important issue, because providing static content in a survivable fashion is relatively simple: burn it to read-only media, and reboot periodically.

Another problem with natural diversity is that it is *expensive*, which is why the IT industry is moving in the opposite direction, consolidating on as few platforms as possible to reduce costs. Heterogeneity costs money for several reasons:

- The site must employ staff with expertise in each of the heterogeneous systems present. Many systems administrators know only a few systems, and those that know many systems tend to be senior and cost more.
- The site must patch each of these systems, and to the extent that vulnerabilities are not common, the patching effort is multiplied by the number of heterogeneous systems present. One study [44] found that a site with an infrastructure of nine NT servers and eight firewalls, for example, would have needed 1315 updates during the first nine months of 2001.
- Multiple versions of applications must be purchased or developed, incurring additional capital and support expenses.

So while heterogeneity can be effective at providing survivability, it comes at a substantial cost. It is not yet clear whether the costs of heterogeneity outweigh the benefits. However, this problem is not specific to the heterogeneity defense; actual survivability is difficult to assess regardless of the methods employed. Section 4 looks at survivability evaluation methods.

4. Evaluating Survivability

The security assurance problem is "How can I tell if this system is secure?" To solve that, one must answer "Will this program do something bad when presented with 'interesting' input?" Unfortunately, to solve that, one must solve Turing's Halting Problem [85], and Turing's theorem proves that you cannot, in general, write a program that will examine arbitrary other programs and their input and determine whether or not they will halt.

Thus in the fully general case, the security assurance problem cannot be statically decided automatically, and so other means must be employed to determine the security assurance of a system. Because determining the *actual* security of systems is so problematic, security standards such as the TCSEC ("Orange Book") and Common Criteria turned instead to documentation of how hard the developers *tried* to provide security, by verifying the inclusion of security enhancing features (access controls, audit logs, etc.) and the application of good software engineering practice (source code control, design documentation, etc.) and at higher levels of certification, a degree of testing.

The question of "How survivable is this system?" is even more problematic, because the survivability question entails assuming bugs in the software, further weakening assumptions on which to base assurance arguments. Section 4.1 looks at studies to evaluate survivability through formal methods, and Section 4.2 looks at empirical evaluations of survivability.

4.1 Formal Methods

Knight et al. [53] investigated the problem of a rigorous definition of "survivability." They concluded a system is "survivable" if it "meets its survivability specification." Since most systems are not formally specified in the first place, and some of the most pressing survivability problems occur in systems largely written using informal languages such as C [49], Knight's survivability metric is not yet of practical use, and there is a great deal of work yet to be done before we can specify just how "survivable" a given system really is.

Similarly, Dietrich and Ryan [30] find that survivability is, by definition, poorly defined, and therefore ill-suited to formal methods of evaluation. They observe that survivability is a relative property, not an absolute, and so must be evaluated in the context of "more survivable than what?"

Gao et al. [39] propose a survivability assessment method that models dependencies of components on one another, with the mission objective as the root. They can then determine which component failures will lead to a failure of the mission. The limitation of this approach, apart from the cost of constructing such a model for large systems, is that for many practical systems, the model would quickly indicate that exploiting a failure in a trusted software component can compromise the mission, that a very large fraction of the software is trusted, and thus the survivability of the system against security attack reduces to the probability of exploitable vulnerabilities in a large software base, which is hard to assess.

Many survivability systems depend critically on the quality of intrusion detection. Tan and Maxion [81] present a thorough statistical analysis that highlights the pitfalls of statistical anomaly detection, especially with respect to the quality of the "good" training data. They show that emergent values from previous anomaly training studies such as Forrest et al.s claim that 4 to 6 events is an optimal window of events to consider [35] are in fact properties of the training data, and not constants at all. This brings into question whether such anomaly training can be effective in the face of intelligent adversaries, as highlighted by Wagner and Soto's Mimicry attack [90].

4.2 Empirical Methods

The limitations of formal methods of assuring survivability make empirical methods of measuring survivability attractive. Unfortunately, for reasons similar to Turing's Halting problem above making static security assurance problematic, it is also not possible to purely test for security. Software testing is comprised of testing probable inputs, random inputs, and boundary conditions. This is inadequate for security testing, because *obscure* input cases can induce security faults and go undetected for many years, as exemplified by `printf` format string vulnerabilities [21]. Just as

Turing shows that you cannot write a static analyzer to determine whether a program will behave badly when given arbitrary input, you cannot test for whether a program will behave badly when given arbitrarily bad input.

So rather than attempt to exhaustively test software for vulnerability, security testing takes the form of measuring the *attacker's work factor*; how much effort must the attacker apply to break the system. *Red team* experimentation (also known as *ethical hackers*) is where a would-be defender deliberately hires an adversarial team to attempt to break into the defender's systems. Red teams and actual attackers use largely the same methods of attack. The critical differences are:

- That red teams can be given boundaries. Defenders can ask red teams to attack only selected subsets of the actual system (e.g., don't attack the production payroll system on payday) and expect these boundaries to be respected.
- That red teams will explain what they have done. Actual attackers may leave an amusing calling card (web site defacement) but they often do not explain the exact details of how they succeeded in compromising security, making it rather expensive to first discover and then repair the holes. Professional red teams, in contrast, will report in detail what they did, allowing defenders to learn from the process.

DARPA funded red team experimentation on the effectiveness of the DARPA-funded survivability technologies. For example, Levin [56] describes several red team experiments testing the validity of a scientific hypothesis. Ideally the experiment should be repeatable, but that is problematic: Levin found that it is important to set the goals of the experiment and the rules of engagement clearly, because undocumented claims and attacks cannot be validated.

The results are very sensitive to what the red team knows about the defender's system at the time of the experiment: if the defender's system is obscured, then the red team will spend most of their time discovering what the system is doing rather than actually attacking it. While this is arguably similar to the situation faced by actual attackers, it is an expensive use of the red team's time, because it can be fairly safely assumed that actual attackers will be able to learn whatever they want about the defender's system. The experimental results also depend heavily on the skills of the red team, which are hard to reproduce exactly: a different team will assuredly have a different set of skills, producing different rates of success against various defenses.

An alternate approach to red team experimentation is symmetric hacker gaming, in which the individual designated attacker and defender teams of a classical red team engagement are replaced by a handful of teams that are competitively set to attack each other while simultaneously defending themselves [20]. Each team is required to

maintain an operational set of network services and applications, and a central score-keeping server records each team's success at keeping services functional, as well as which team actually "owns" a given server or service. This symmetric threat model tends to reduce disputes over the rules of engagement, because all teams are *equally* subject to those rules. This results is a nearly no-holds-barred test of survivability.

We entered an Immunix server in the Defcon "Root Fu" (nee "Capture the Flag") games in 2002 and 2003, with mixed results. In both games, we placed 2nd of 8 teams. In the 2002 game, the Immunix machine was never compromised, but it did take most of the first day to configure the Immunix secure OS such that it could earn points, because the required functionality was obscured, being specified only as a reference server image that did provide the required services, but was also highly vulnerable. However, the Immunix server was also DoS'd to the point where it no longer scored points; in retrospect not very surprising, as Immunix was designed to prevent intrusion, not DoS. The 2003 game explicitly prohibited DoS attacks, but teams deployed DoS attacks anyway. So even in symmetric red teaming, rules of engagement matter, if the rules are not uniformly enforced.

Another form of empirical testing is to measure the precision of intrusion detection using a mix of test data known to be either "good" or "bad." DARPA sponsored such a study in the 1998 MIT/Lincoln Labs intrusion detection test [57,61]. The goal of DARPA's intrusion detection research was to be able to detect 90% of attacks (including especially *novel* attacks) while reducing false positive reports by an order of magnitude over present intrusion detection methods. Intrusion detection systems deployed by the US Air Force in the mid-1990s were expensive to operate because network analysts had to spend many hours investigating false positive "intrusion events" that were actually benign.

The results were mixed. False positive rates were significantly lowered versus previous generations of technologies, but were still high enough that intrusion detection still requires significant human intervention. Worse, the detectors failed to detect a significant number of novel attacks. Only a few of the tested technologies were able to detect a few of the novel attacks.

Yet another aspect of empirical measurement is to examine the behavior of attackers. This behavior matters because survivability is essentially fault tolerance against the faults that attackers will induce, and so expectations of survivability need to be measured against this threat. Browne, Arbaugh, McHugh and Fithen [13] present a trend analysis of exploitation, studying the rates at which systems are compromised, with respect to the date on which the vulnerabilities in question were made public. This study showed that exploitation spikes not immediately following *disclosure* of the vulnerability, but rather after the vulnerability is *scripted* (an automatic exploit is written and released).

Our own subsequent study [7] statistically examined the relative risks of patching early (risk of self-corruption due to defective patches) versus the risk of patching later (risk of security attack due to an unpatched vulnerability) and found that approximately 10 days after a patch is released is the optimal time to patch. However, the gap between the time a vulnerability is disclosed and when it is scripted appears to be closing rapidly [91] and so this number is expected to change.

5. Related Work

The field of Information Survivability dates back to the early 1990s, when DARPA decided to take a fresh approach to the security problem. In a field so young, there are not many survey papers. In 1997, Ellison et al. [31] surveyed this emerging discipline, which they characterize as the ability of a system to carry out its mission while connected to an *unbounded network*. An unbounded network is one with no centralized administration, and thus attackers are free to connect and present arbitrary input to the system, exemplified by the Internet. They distinguish survivability from security in that survivability entails a capacity to recover. Unfortunately, they were *anticipating* the emergence of recovery capability, and that capability has yet to effectively emerge from survivability research. Self recovery capacity remains an area of strong interest [59]. They distinguish survivability from fault tolerance in that fault tolerant systems make failure statistically improbable in the face of random failures, but cannot defend against coincident failures contrived by attackers, as described in Section 2.

Stavridou et al. [77] present an architectural view of how to apply the techniques of fault tolerance to provide intrusion tolerance. They propose that individual components should be sufficiently simple that their security properties can be formally assured, and that the entire system should be multilevel secure (so security faults are isolated) as in Section 3.1.

Powell et al. [1] describe the MAFTIA (Malicious- and Accidental-Fault Tolerance for Internet Applications) conceptual model and architecture. This is a large document with 16 authors, describing a long-term project. They define *dependability* as the ability to deliver service that can justifiably be trusted, *survivability* as the capability of a system to fulfill its mission in a timely manner, and *trustworthiness* as assurance that a system will perform as expected. They conclude that all three of these concepts are essentially equivalent. Dependability has been studied for the last thirty years by organizations such as the IFIP working group 10.4, and from this perspective, survivability can be viewed as a relatively recent instance of dependability studies.

In 2000 we surveyed *post hoc* security enhancement techniques [25] which subsequently came to be known as intrusion prevention. This survey considered *adaptations* (enhancements) in two dimensions:

- *What* is adapted:
  - *Interface*: the enhancement changes the interface exposed to other components.
  - *Implementation*: the enhancement is purely internal, nominally not affecting how the component interacts with other components.
- *How* the enhancement is achieved:
  - *Restriction*: the enhancement *restricts* behavior, either through misuse detection or anomaly detection (see Section 3.3).
  - *Randomization*: the enhancement uses natural or synthetic diversity to randomize the system so as to make attacks non-portable with respect to the defender's system (see Section 3.4).

This two-dimensional space thus forms quadrants. We found that effective techniques exist in all four quadrants, but that in most cases, restriction is more *cost-effective* than randomization. Interestingly, we found that it is often the case that when one goes looking for a randomization technique, one finds a restriction technique sitting conceptually beside the randomization technique that works better: if a system attribute can be identified as something the attacker depends on, then it is better to restrict the attacker's access to that resource than to randomize the resource.

We also conducted a similar study with narrower focus, examining buffer overflow attacks and defenses [28]. Responsible for over half of all security faults for the last seven years, buffer overflows require special attention. A buffer overflow attack must first arrange for malicious code to be present in the victim process's address space, and then must induce the victim program to transfer program control to the malicious code. Our survey categorized attacks in terms of how these objectives can be achieved, defenses in terms of how they prevent these effects, and summarized with effective combinations of defenses to maximize coverage.

6. Conclusions

Reliability methods (redundancy, fault injection, etc.) provide protection against independent faults. Security methods (least privilege, type checking, etc.) defend against malicious exploitation of software design and implementation defects.

Survivability methods combine these two techniques to provide survivability of systems against combinations of accidental and malicious faults, and also to defend

against failures in security systems. Survivability techniques defend against security faults occurring at various times throughout the software life cycle, from design time, to implementation time, to run time, to recovery time.

REFERENCES

[1] Adelsbach A., Cachin C., Creese S., Deswarte Y., Kursawe K., Laprie J.-C., Powell D., Randell B., Riodan J., Ryan P., Simmionds W., Stroud R.J., Verssimo P., Waidner M., Wespi A., *Conceptual Model and Architecture of MAFTIA*, LAAS-CNRS, Toulouse, and University of Newcastle upon Tyne, January 31, 2003. Report MAFTIA deliverable D21, http://www.laas.research.ec.org/maftia/deliverables/D21.pdf.

[2] Arce I., "Woah, please back up for one second", http://online.securityfocus.com/archive/98/142495, October 31, 2000. Definition of security and reliability.

[3] Badger L., Sterne D.F., et al., "Practical domain and type enforcement for UNIX", in: *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA*, May 1995.

[4] Badger L., Sterne D.F., Sherman D.L., Walker K.M., Haghighat S.A., "A domain and type enforcement UNIX prototype", in: *Proceedings of the USENIX Security Conference*, 1995.

[5] Balzer R., "Assuring the safety of opening email attachments", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.

[6] Baratloo A., Singh N., Tsai T., "Transparent run-time defense against stack smashing attacks", in: *2000 USENIX Annual Technical Conference, San Diego, CA*, June 18–23, 2000.

[7] Beattie S.M., Cowan C., Arnold S., Wagle P., Wright C., Shostack A., "Timing the application of security patches for optimal uptime", in: *USENIX 16th Systems Administration Conference (LISA), Philadelphia, PA*, November 2002.

[8] Bellovin S.M., "Distributed firewalls", *;login:* **24** (November 1999).

[9] Bester J., Walther A., Erlinger M., Buchheim T., Feinstein B., Mathews G., Pollock R., Levitt K., "GlobalGuard: Creating the IETF-IDWG Intrusion Alert Protocol (IAP)", in: *DARPA Information Survivability Conference Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.

[10] Bhatkar S., DuVarney D.C., Sekar R., "Address obfuscation: an approach to combat buffer overflows, format-string attacks, and more", in: *12th USENIX Security Symposium, Washington, DC*, August 2003.

[11] Bobert W.E., Kain R.Y., "A practical alternative to hierarchical integrity policies", in: *Proceedings of the 8th National Computer Security Conference, Gaithersburg, MD*, 1985.

[12] Bray B., Report, *How Visual C++ .Net Can Prevent Buffer Overruns*, Microsoft, 2001.

[13] Browne H.K., Arbaugh W.A., McHugh J., Fithen W.L., "A trend analysis of exploitations", in: *Proceedings of the 2001 IEEE Security and Privacy Conference, Oakland, CA*, May 2001, pp. 214–229, http://www.cs.umd.edu/~waa/pubs/CS-TR-4200.pdf.

[14] CERT Coordination Center, “CERT Advisory CA-1999-04 Melissa Macro Virus”, http://www.cert.org/advisories/CA-1999-04.html, March 27, 1999.
[15] CERT Coordination Center, “CERT Advisory CA-2000-04 Love Letter Worm”, http://www.cert.org/advisories/CA-2000-04.html, May 4, 2000.
[16] CERT Coordination Center, “CERT Advisory CA-2002-07 Double Free Bug in zlib Compression Library”, http://www.cert.org/advisories/CA-2002-07.html, March 12, 2002.
[17] CERT Coordination Center, “CERT Incident Note IN-2003-03”, http://www.cert.org/incident_notes/IN-2003-03.html, August 22, 2003.
[18] Chen H., Wagner D., “MOPS: an infrastructure for examining security properties of software”, in: *Proceedings of the ACM Conference on Computer and Communications Security, Washington, DC*, November 2002.
[19] Cheswick W.R., Bellovin S.M., *Firewalls and Internet Security: Repelling the Wily Hacker*, Addison-Wesley, 1994.
[20] Cowan C., Arnold S., Beattie S.M., Wright C., “Defcon capture the flag: Defending vulnerable code from intense attack”, in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.
[21] Cowan C., Barringer M., Beattie S., Kroah-Hartman G., Frantzen M., Lokier J., “FormatGuard: automatic protection from printf format string vulnerabilities”, in: *USENIX Security Symposium, Washington, DC*, August 2001.
[22] Cowan C., Beattie S., Johansen J., Wagle P., “PointGuard: protecting pointers from buffer overflow vulnerabilities”, in: *USENIX Security Symposium, Washington, DC*, August 2003.
[23] Cowan C., Beattie S., Pu C., Wagle P., Gligor V., “SubDomain: parsimonious server security”, in: *USENIX 14th Systems Administration Conference (LISA), New Orleans, LA*, December 2000.
[24] Cowan C., Beattie S., Wright C., Kroah-Hartman G., “RaceGuard: kernel protection from temporary file race vulnerabilities”, in: *USENIX Security Symposium, Washington, DC*, August 2001.
[25] Cowan C., Hinton H., Pu C., Walpole J., “The cracker patch choice: an analysis of post hoc security techniques”, in: *Proceedings of the 19th National Information Systems Security Conference (NISSC 2000), Baltimore, MD*, October 2000.
[26] Cowan C., Pu C., Hinton H., “Death, taxes, and imperfect software: surviving the inevitable”, in: *Proceedings of the New Security Paradigms Workshop, Charlottesville, VA*, September 1998.
[27] Cowan C., Pu C., Maier D., Hinton H., Bakke P., Beattie S., Grier A., Wagle P., Zhang Q., “StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks”, in: *7th USENIX Security Conference, San Antonio, TX*, January 1998, pp. 63–77.
[28] Cowan C., Wagle P., Pu C., Beattie S., Walpole J., “Buffer overflows: attacks and defenses for the vulnerability of the decade”, in: *DARPA Information Survivability Conference and Expo (DISCEX)*, January 2000. Also presented as an invited talk at SANS 2000, March 23–26, 2000, Orlando, FL, http://schafercorp-ballston.com/discex.
[29] “ “Solar Designer”, Non-Executable User Stack”, http://www.openwall.com/linux/.

[30] Dietrich S., Ryan P.Y.A., "The survivability of survivability", in: *Proceedings of the Information Survivability Workshop (ISW 2002), Vancouver, BC*, March 2002.
[31] Ellison R.J., Fisher D.A., Linger R.C., Lipson H.F., Longstaff T., Mead N.R., *Survivable Network Systems: An Emerging Discipline, Report CMU/SEI-97-TR-013*, Software Engineering Institute, November 1997, http://www.cert.org/research/tr13/97tr013title.html.
[32] Eskin E., Lee W., Stolfo S.J., "Modeling system calls for intrusion detecting with dynamic window sizes", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.
[33] Etoh H., "GCC extension for protecting applications from stack-smashing attacks", http://www.trl.ibm.com/projects/security/ssp/, November 21, 2000.
[34] Feinstein L., Schnackenberg D., Balupari R., Kindred D., "Statistical approaches to DDoS attack detection and response", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.
[35] Forrest S., Hofmeyr S.A., Somayaji A., Longstaff T.A., "A sense of self for UNIX processes", in: *Proceedings of the IEEE Symposium on Security Privacy, Oakland, CA*, 1996.
[36] Forrest S., Somayaji A., Ackley D.H., "Building diverse computer systems", in: *HotOS-VI*, May 1997.
[37] Fraiser T., Loscocco P., Smalley S., et al., "Security enhanced Linux", http://www.nsa.gov/selinux/, January 2, 2001.
[38] Fraser T., Badger L., Feldman M., "Hardening COTS software with generic software wrappers", in: *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA*, May 1999.
[39] Gao Z., Hui Ong C., Kiong Tan W., "Survivability assessment: modelling dependencies in information systems", in: *Proceedings of the Information Survivability Workshop (ISW 2002), Vancouver, BC*, March 2002.
[40] Ghosh A.K., Schwatzbard A., Shatz M., "Learning program behavior profiles for intrusion detection", in: *Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA*, April 1999.
[41] Gosling J., McGilton H., "The Java language environment: A White Paper", http://www.javasoft.com/docs/white/langenv/, May 1996.
[42] Hamonds K.H., "The strategy of the Fighter Pilot", *Fast Company* **59** (June 2002).
[43] Hollebeek T., Berrier D., "Interception, wrapping and analysis framework for Win32 Scripts", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.
[44] Hurley E., "Keeping up with patch work near impossible", SearchSecurity, http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci796744,00.html, January 17, 2002.
[45] "Tripwire Security Incorporated. Tripwire.org: Tripwire for Linux", http://www.tripwire.org/.
[46] Jim T., Morrisett G., Grossman D., Hicks M., Cheney J., Wang Y., "Cyclone: A safe dialect of C", in: *Proceedings of USENIX Annual Technical Conference, Monteray, CA*, June 2002.

[47] Just J.E., Reynolds J.C., "HACQIT: Hierarchical adaptive control of QoS for intrusion tolerance", in: *Annual Computer Security Applications Conference (ACSAC), New Orleans, LA*, December 10–14, 2001.

[48] Kc G.S., Keromytis A.D., Prevelakis V., "Countering CodeInjection attacks with InstructionSet randomization", in: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS 2003), Washington, DC*, October 2003.

[49] Kernighan B.W., Ritchie D.M., *The C Programming Language*, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1988.

[50] Kewley D., Fink R., Lowry J., Dean M., "Dynamic approaches to thwart adversary intelligence gathering", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.

[51] Kim G.H., Spafford E.H., "Writing, supporting, and evaluating Tripwire: A publicly available security tool", in: *Proceedings of the USENIX UNIX Applications Development Symposium, Toronto, Canada*, 1994, pp. 88–107.

[52] Knight J.C., Leveson N.G., "An experimental evaluation of the assumptions of independence in multiversion programming", *IEEE Transactions on Software Engineering* **12** (1) (1986) 96–109.

[53] Knight J.C., Strunk E.A., Sullivan K.J., "Towards a rigorous definition of information system survivability", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.

[54] Lampson B.W., "Protection", in: *Proceedings of the 5th Princeton Conference on Information Sciences and Systems, Princeton, NJ*, 1971. Reprinted in *ACM Operating Systems Review* **8** (1) (January 1974) 18–24.

[55] Lee W., Stoflo S.J., Chan P.K., Eskin E., Fan W., Miller M., Hershkop S., Zhang J., "Real time data mining-based intrusion detection", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.

[56] Levin D., "Lessons learned in using live red teams in IA experiments", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.

[57] Lippmann R., Haines J.W., Fried D.J., Korba J., Das K., "The 1999 DARPA off-line intrusion detection evaluation", in: *Recent Advances in Intrusion Detection (RAID), Toulouse, France*, October 2–4, 2000.

[58] Liu P., "Engineering a distributed intrusion tolerant database system using COTS components", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.

[59] Liu P., Pal P., *Workshop on Survivable and Self-Regenerative Systems*, October 31, 2003. In conjunction with the ACM International Conference on Computer and Communications Security (CCS-10).

[60] Loscocco P., Smalley S., "Integrating flexible support for security policies into the Linux operating system", in: *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01)*, June 2001.

[61] McHugh J., "The 1998 Lincoln Lab IDS evaluation—a critique", in: *Recent Advances in Intrusion Detection (RAID), Toulouse, France*, October 2–4, 2000.

[62] Michael C.C., "Finding the vocabulary of program behavior data for anomaly detection", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.

[63] Milner R., Tofte M., Harper R., *The Definition of Standard ML*, The MIT Press, 1990.

[64] Necula G.C., McPeak S., Weimer W., "CCured: type-safe retrofitting of legacy code", in: *Proceedings of the 29th ACM Symposium on Principles of Programming Languages (POPL02), London, UK*, January 2002. Also available at http://raw.cs.berkeley.edu/Papers/ccured_popl02.pdf.

[65] O'Brien D., "Intrusion tolerance via network layer controls", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.

[66] Papadopoulos C., Lindell R., Mehringer J., Hussain A., Govindan R., "COSSACK: Coordinated Suppression of Simultaneous Attacks", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.

[67] Payne C., Markham T., "Architecture and applications for a distributed embedded firewall", in: *Annual Computer Security Applications Conference (ACSAC), New Orleans, LA*, December 10–14, 2001.

[68] Porras P., Neumann P., "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", in: *Proceedings of the 20th National Information Systems Security Conference (NISSC 1997), Baltimore, MD*, October 1997.

[69] Ptacek T.H., Newsham T.N., *Insertion, Evation, and Denial of Service: Eluding Network Intrusion Detection, Report*, Network Associates Inc., January 1998, http://www.nai.com/products/security/advisory/papers/ids-html/doc001.asp.

[70] Rushby J., "Critical system properties: Survey and taxonomy", *Reliability Engineering and System Safety* **43** (2) (1994) 189–219.

[71] Saltzer J.H., Schroeder M.D., "The protection of information in computer systems", *Proceedings of the IEEE* **63** (9) (November 1975).

[72] Savage S., Wetherall D., Karlin A., Anderson T., "Network support for IP traceback", *IEEE/ACM Transactions on Networking* **9** (3) (June 2001) 226–237.

[73] Schmid M., Hill F., Ghosh A.K., Bloch J.T., "Preventing the execution of unauthorized Win32 applications", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.

[74] Schnackenberg D., Djahandari K., Sterne D., "Infrastructure for intrusion detection and response", in: *DARPA Information Survivability Conference and Expo (DISCEX)*, January 2000.

[75] Secure Software, "RATS: Rough Auditing Tool for Security", http://www.securesoftware.com/download_rats.htm, July 2002.

[76] Song D., "Fragroute", http://monkey.org/~dugsong/fragroute/, May 27, 2002.

[77] Stavridou V., Dutertre B., Riemenschneider R.A., Saldi H., "Intrusion tolerant software architectures", in: *DARPA Information Survivability Conference and Expo (DISCEX II), Anaheim, CA*, June 12–14, 2001.

[78] Strom R.E., Bacon D.F., Goldberg A., Lowry A., Yellin D., Yemini S.A., *Hermes: A Language for Distributed Computing*, Prentice-Hall, 1991.

[79] Strom R.E., Yemini S.A., "Typestate: A programming language concept for enhancing software reliability", *IEEE Transactions on Software Engineering* **12** (1) (January 1986) 157–171.
[80] Stroustrup B., *The C++ Programming Language*, Addison-Wesley, Reading, MA, 1987.
[81] Tan K.M.C., Maxion R.A., "'Why 6?" Defining the operational limits of stide, an anomaly-based intrusion detector", in: *Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA*, May 2002.
[82] "'The PaX Team'. PaX", http://pageexec.virtualave.net/, May 2003.
[83] "'tf8", Wu-Ftpd remote format string stack overwrite vulnerability", http://www.securityfocus.com/bid/1387, June 22, 2000.
[84] Thomas R., Mark B., Johnson T., Croall J., "NetBouncer: client-legitimacy-based high-performance DDoS filtering", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.
[85] Turing A., "On computable numbers with an application to the Entscheidungsproblem", *Proc. London Math. Society* **42** (2) (1937) 230–265.
[86] Valdes A., "Detecting novel scans through pattern anomaly detection", in: *DARPA Information Survivability Conference and Expo (DISCEX III), Washington, DC*, April 22–24, 2003.
[87] Viega J., Bloch J.T., Kohno T., McGraw G., "ITS4: A static vulneability scanner for C and C++ code", in: *Annual Computer Security Applications Conference (ACSAC), New Orleans, LA*, December 2000, http://www.cigital.com/its4/.
[88] Vigna G., Eckmann S.T., Kemmerer R.A., "The STAT tool suite", in: *DARPA Information Survivability Conference and Expo (DISCEX)*, January 2000.
[89] Wagner D., Foster J.S., Brewer E.A., Aiken A., "A first step towards automated detection of buffer overrun vulnerabilities", in: *NDSS (Network and Distributed System Security), San Diego, CA*, February 2000.
[90] Wagner D., Soto P., "Mimicry attacks on HostBased intrusion detection systems", in: *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC*, October 2002.
[91] Walsh L.M., "Window of opportunity closing for patching", *Security Wire Digest* **5** (66) (September 4, 2003), http://infosecuritymag.techtarget.com/ss/0,295812,sid6_iss82,00.html#news2.
[92] Wheeler D., "Flawfinder", http://www.dwheeler.com/flawfinder/, July 2, 2002.
[93] Wright C., Cowan C., Smalley S., Morris J., Kroah-Hartman G., "Linux security module framework", in: *Ottawa Linux Symposium, Ottawa, Canada*, June 2002.
[94] Wright C., Cowan C., Smalley S., Morris J., Kroah-Hartman G., "Linux security modules: general security support for the Linux kernel", in: *USENIX Security Symposium, San Francisco, CA*, August 2002, http://lsm.immunix.org.
[95] Xu J., Kalbarczyk Z., Iyer R.K., "Transparent runtime randomization for security", in: *Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS'2003), Florence, Italy*, October 2003.
[96] Zhang Y., Dao S.K., Vin H., Alvisi L., Wenke Lee L.A., "Heterogeneous networking: a new survivability paradigm", in: *Proceedings of the New Security Paradigms Workshop, Cloudcroft, NM*, September 2001.

Smart Cards

KATHERINE M. SHELFER

Drexel University, Philadelphia, PA, USA
kathy.shelfer@xis.drexel.edu

CHRIS CORUM

Avisian, Corp., Tallahassee, FL, USA
chris@avisian.com

J. DREW PROCACCINO

Rider University, Lawrenceville, NJ, USA
jdproc@aol.com

JOSEPH DIDIER

Infinacard, Inc., St. Petersburg, FL, USA
jdider@infinacard.com

Abstract

This paper presents an overview of the history, commercialization, technology, standards, and current and future applications of smart cards. Section 1 is an overview of smart cards, including their current global use in *identification*, *verification* and *authorization* applications through their ability to support transaction processing, information management and multiple applications on a single card. This section also includes a summary of the invention and early development and application of smart cards. The second section describes a typical smart card-based transaction, tracing it from the initial contact between a card and the card reader through the transaction to termination of the transaction. The third section describes the physical characteristics of the smart card, and its associated contact and contactless interfaces, integrated circuit (IC) chip and processor capacity. Section 4 summarizes the international standards associated with smart cards, including those related to interoperability among contact and contactless

ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60005-7

cards, and their respective reading devices. In Section 5, the focus is a high-level discussion of associated access technologies, including a more detailed look at magnetic stripe and barcode technologies and standards. This section includes a very brief mention of the impact of RISC-based technologies and Sun's Java™ Virtual Machine®. Section 6 discusses smart card security relating to the card's ability to authorize and facilitate *electronic*, *logical* and *physical access* to controlled applications and physical locations. Also discussed is *physical security*, which relates to cardholders, environment and cards tampering, and *data security*, which is related to smart cards ability to support cryptography and cross validation of data stored on the cards across multiple databases for purposes of identification verification. Section 7 concludes this paper with a look at the future of smart card-related developments, including those related to both *technology* and *applications*. Technology-related developments include the support of more than a single operating system on the processor chip and peripheral card technologies. Application-related developments include those related to identification, information storage and transaction processing.

1. Introduction

1.1 Overview

At this time, smart card applications are used to (1) encourage and protect lawful economic activity; (2) ensure the survival of critical infrastructures and (3) protect individuals and societies from those who would deliberately do them harm. There are many contributing factors that determine the development and deployment of smart card systems. Among these are the way in which smart cards were invented and commercialized; current directions in applied research and development; the development and support of international standards; and the impact of human concerns about data security and data privacy. Any item with an embedded microprocessor chip can be considered "smart," from keychain fobs to cooking utensils [9]. The most familiar form, however, is a thin plastic card (refer to Section 3, Smart card technology) that contains one or more integrated circuit (IC) chips. The microprocessor, or "smart" chip, interacts either physically with (contact) or in proximity to (contactless) a smart card reader, or both. Information on the card interacts with information in the reader to authorize the requested session (transaction), for which the chip has been programmed [21,74].

The smart card is designed to store and serve a range of personal data related to the cardholder that is used to authorize specific transactions. The value of the smart card is that it can be used to secure digital transactions that rely on personal *identification*, identity *verification*, and transaction *authorization* [40,82].

- *Identification*. Smart cards store information about the cardholder's identity in digital form—bank account numbers, organizational affiliations, personal biometrics, etc.—that is used to secure digital transactions, i.e., ATM transactions and eCommerce [57,58].
- *Verification*. The smart card stores a range of personal identity data, i.e., biometrics, that provide means of comparing digital identities with physical identities in settings where neither form of identification would be sufficient.
- *Authorization.* Smart cards are used to electronically authorize the cardholder's right to initiate and engage in specific transactions that involve logical, physical and electronic access controls.

Smart cards allow 2-party digital transactions (cardholder and card issuer) to be documented by a 3rd party—in this case, the smart card system. Smart cards support one or more of the following applications:

(1) *Credit.* Smart cards secure merchant transactions and reduce fraud. This allows cardholders to have financial flexibility in the form of pre-approved transborder cash advances and loans, initiated at the point-of-sale that are tied directly to the purchase of specific goods/services. The credit function rarely requires a personal identification number.

(2) *Debit*. Smart cards are used to provide electronic direct debit at ATM machines and at specific points-of-sale. The debit function improves the cardholder's access to specific goods and services in locations where the cardholder's identify and credit history are unknown, and protects the cardholder's banking relationship by allowing a 3rd-party to directly debit the cardholder's financial institution for the purchase of specified goods/services at the point of sale. It should be noted that such transactions in theory—but not always in practice—require a personal identification number (PIN) as an additional layer of security. This situation is currently being discussed in contract negotiations and in the courts.

(3) *Stored-value*. A fixed value is initially encoded and subsequent purchases are deducted (nominally, but not always, to a "zero" balance). This transaction does not require access to the cardholder's personal identity, so no PIN is required. Two examples of single-function stored-value cards are telephone cards and retail merchant gift cards. On magnetic stripe cards, this stored value is placed on a "junk" stripe. On smart cards, this stored value is placed in one or more electronic "purses." Stored-value may be disposable (debit "read-only") or re-loadable (debit/credit "read-write"). While ownership of the abandoned value on the card, or *escheat*, is disputable, it has been treated as a source of substantial additional revenue in some settings, including the Olympic games held in Atlanta [30,32,63,72].

(4) *Information management*. Card issuers provide these cards to individuals in order to facilitate the portable storage and use of the cardholder's personal information, i.e., bank credit/debit accounts, insurance information, medical history, emergency contact information and travel documents. http://www.iso.org/iso/en/commcentre/isobulletin/articles/2003/pdf/medicalcard03-06.pdf. This information is used to verify the cardholder's identity and authorize specified digital transactions [8,60,66,69,77–79,91].

(5) *Loyalty/affinity*. A variety of vendor incentives, recorded at the point of sale, are tied to the purchase of goods/services. Incentives may include points, credits, discounts and/or direct delivery of products/services. Since the cardholder's identity is known, purchasing patterns can be tracked and loyal cus-

tomers can select their own rewards (http://www.bai.org/bankingstrategies/2002-may-jun/smart/). One web site claims that loyalty programs are important to 64% of USA households, and used by over 50% of Americans (http://www.jmac-solutions.com/giftcards/topic_3.asp). Clearly identifiable personal data has been leased/sold to generate additional (cross-marketing/co-branding) revenue for some time, but smart cards have the potential to make more data available for analysis. It should be noted that ownership and use of data that clearly identifies individuals and their behavior patterns is disputable.

(6) *Multi-application*. The ability to combine one or more applications on a single card (some of which are able to share PIN numbers) requires the capabilities associated with the integrated circuit on the smart card [28]. While forecasts vary considerably based on the source, shipments of multi-application cards have increased significantly since 1998, with roughly half of these being Java cards (http://developers.sun.com/techtopics/mobility/javacard/) [92].

1.2 The Invention of Smart Cards

The earliest smart card patents were mainly theoretical in nature, as the technology to take advantage of this innovative thinking was not actually available until 1976 and later. The earliest credit for smart card-*related* patents should probably be awarded to Jürgen Dethloff and Helmut Grötrupp, two German inventors who patented the idea of having plastic cards hold microchips in 1968 [42]. In 1970, Dr. Kunitaka Arimura, a Japanese inventor, patented concepts related to the IC card [1]. However, Mr. Roland Moreno, a French finance and technology journalist, is actually credited with being the "father" of today's smart card based on his 1974 patent on chip cards [59].

Dr. Arimura limited his patents to Japan, where he controlled access to that technology through the Arimura institute. Mr. Moreno, however, founded Innovatron to exploit his patent on a larger scale. The first Innovatron license, granted in 1976 to Honeywell Bull, a French computer manufacturer, was worldwide in scope. Bull, a pioneer in information technology, was the first company in the world to actually make substantial progress [68]. Bull contracted with a US company, Motorola Semiconductor, to develop the necessary microprocessor. This innovation was introduced in 1977 [34] and a functioning chip card was introduced in 1979. Known as the CP8, an acronym for *Portable Computer for the 80s*, the name stuck. The first Bull smart card division was named CP8 Transac. In 1979, Schlumberger, a French petroleum company, obtained a worldwide license and the Dutch electronics company Philips obtained rights for France (only). Each company began independent development of cards and terminals. The Bull smart card product line generated up to ten

percent of Groupe Bull's revenue prior to its purchase by Schlumberger Limited's Test and Transactions division. On its corporate web site (http://www.axalto.com), Axalto (formerly known as SchlumbergerSema's smart card division) cites Gartner, 2003 and Frost & Sullivan 2003 in support of its claim to be *the world's largest supplier and leading innovator, with* 4500 *employees in more than* 100 *countries and worldwide sales of more than* 2.8 *billion smart cards to date.*

Credit for the commercial success of the technology itself [11] should probably be awarded to France. Beginning in 1976, France's Directorate General of Telecommunications (DGT) established a five-year initiative, the Telematique Programme, to rebuild France's pay telephone infrastructure and encourage growth in the country's lagging computer industry. This initiative emphasized two technologies: (1) *videotext* (later Minitel, a prophetic forerunner of the Internet); and (2) *smart cards*, envisioned even then as an identification and payment tool linked to computer communication. France Télécom issued *smart card readers that made it possible to purchase everything from opera tickets to train reservations online—well before anyone had heard of the Internet* [59]. The French government quickly identified additional economic benefits, such as those associated with the elimination of coins and/or tokens in public telephone booths. This led France to introduce smart pre-paid telephone cards in the 1980s [1]. While Europe continues to hold the lead, today there are other countries, including the USA, that also offer some form of card-based access to telephone services.

Smart cards were first considered to be simply a better way to deal with economic crime, which was one of the driving forces behind the rapid development and launch of smart cards. Cartes Bancaires, the French banking association, found that criminals were illegally scanning traditional magnetic stripes, and then copying this data to counterfeit credit cards [34]. At that time, the regulatory climate favored wireless telecommunications. As a result, most transactions were processed offline, including those in automated teller machines (ATMs). This situation left banks open to ATM fraud, so France's major banks created a study group, the Groupement d'Intérêt Economique (GIE) that was tasked to examine ways to reduce fraud. In 1980, the group issued a national request for proposal (RFP) for smart card technology and the banks in France chose to issue smart cards [59]. Each of the three initial smart card licensees—Bull, Schlumberger, and Philips—tested their products in a point of sale (POS) trial in a separate French city. France's first tests of smart cards in 1980 were not initially successful for several reasons: (1) the card itself was too expensive; (2) card quality was unreliable; and (3) the necessary technical infrastructure to utilize and integrate with the cards was still not available [75,81].

Unlike Europe, the regulatory climate in the USA favored wired communication, so there was not the same degree of urgency to adopt more stringent forms of identity verification. This resulted in a much slower migration to smart cards. In 1982,

two small trials were conducted in the US: (1) the First Bank Systems of Minneapolis trial, involving ten (10) North Dakota farmers, and (2) a Department of Defense (DOD) trial with a few hundred soldiers in Fort Lee, New Jersey. In 1983, Innovatron awarded development rights for the USA, the UK and Japan to Smart Card International, a USA company. In 1986, MasterCard tested the technology in Columbia, Maryland and Palm Beach, Florida.

In addition to smart card innovations themselves, universities have also played a role as early adopters of smart card systems. In the USA, there were a number of college and university campuses that upgraded from magnetic identification cards to smart identification cards. Bill Norwood and eight colleagues left The Florida State University (FSU) to found Cybermark, LLC, a joint venture of Sallie Mae, The Batelle Memorial Institute, the Huntingdon Bank and the Florida State University. As a result of first-hand knowledge of the needs of the target customer group, Cybermark had a first-mover advantage in the design and deployment of campus smart card installations. In May 1999, for example, Cybermark added a credit card option to the card (http://www.autm.net/pubs/survey/1999/cybermark.html). At one point, some sixty people (45 in Tallahassee, Florida) processed transactions for over 700,000 issued cards. The number of cards doubled when SchlumbergerSema asked Cybermark to service its existing campus card installations [17,20,23]. Table I provides a summary timeline of the early history of smart cards.

By 2002, there were still fewer than 35 (mainly European) companies in control of 95% of the smart card market. At one point, three of Europe's top ten high technology companies had smart card product lines, although most of these products

TABLE I
SMART CARD TIMELINE

Year	Event
1968	2 German inventors, Jorgen Dethloff and Helmut Grotrupp, patent the idea of combining micro chips with plastic cards[a]
1970	A Japanese inventor, Kunitaka Arimura receives Japanese (only) patent protection[b]
1974	French journalist, Roland Moreno, receives patent in France[b]
1976	French Licenses awarded to Bull (France) as result of DGT initiative[b]
1979	Schlumberger (France) and Philips (Netherlands) receive first Innovatron licenses[c]
1980	GIE sponsors first smart card trials in 3 French cities[b]
1982	First USA trials held in North Dakota and New Jersey[b]
1996	First USA university campus deploys smart cards[b]
1998	Gemplus introduces contactless IC

[a][42];
[b][81];
[c][3].

TABLE II
SMART CARD SHIPMENTS[a] BY REGION

Region	1999	2000	2001	2002	2003	2004	Totals	Region (%)
North America	21	36	62	117	187	276	699	12
Americas	7	20	39	62	108	151	387	6
Europe	318	398	467	565	687	794	3229	54
Japan	10	23	44	68	103	136	384	6
Asia Pacific	75	121	179	245	313	401	1334	22
Totals	431	598	791	1057	1398	1758	6033	100

[a] Source: Dataquest [67].

still had limited functionality. The smart identification card market itself, however, was not living up to its early promise. This changed as a result of the bombing of the World Trade Center on September 11, 2001. Identity became inextricably linked with protection of national economies, critical infrastructures, and citizens. At the same time, quality had improved, costs of smart card systems had plummeted, and there were many new applications that could take advantage of the chip—most notably, biometrics. This situation resulted in a race to deploy smart card identification systems, especially for government, military and law enforcement personnel and transit authorities [61,64,88]. In addition, there continues to be a growing awareness of tangible and intangible benefits associated with the use of intelligent identification cards [81,55]. For example, a simple Google® search can return thousands of hits on the topic.

Today, the market is truly global. Table II shows recent and predicted worldwide growth of smart card shipments by region [67]. Sadly, like so many other entrepreneurial ventures, Cybermark later failed for business reasons. However, it deserves its place in history as an innovator in multi-application smart card systems. Today, many companies have entered this market, and there are hundreds of companies, several organizations and specialized publications that demonstrate the increasing importance of smart card systems around the world (http://www.smartcardalliance.org; http://www.eurosmart.com).

2. A Typical Smart Card Transaction

As with magnetic cards, the typical IC card transaction, or "session" typically includes five stages: (1) *Contact*; (2) *card validation*; (3) *establishing communication* between card and reader/terminal; (4) *transaction*; and (5) *session termination.* These stages are shown in Fig. 1 and discussed below. A flow diagram of a typical point-of-sale (POS) transaction between a cardholder and a merchant is shown in Fig. 2.

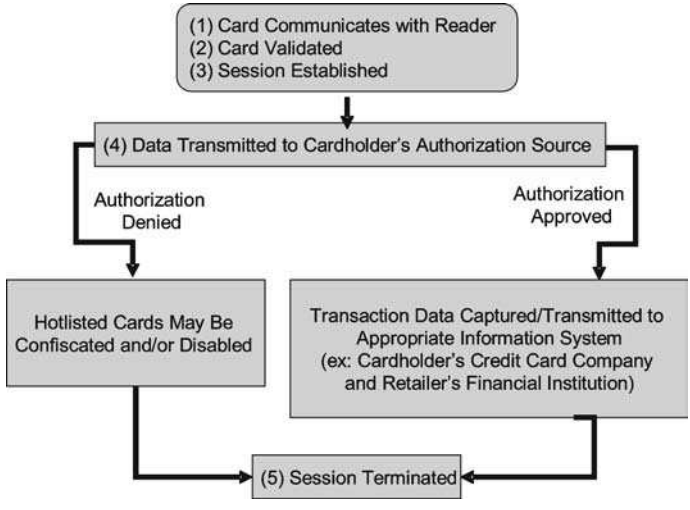


FIG. 1. Data flow of smart card transaction.

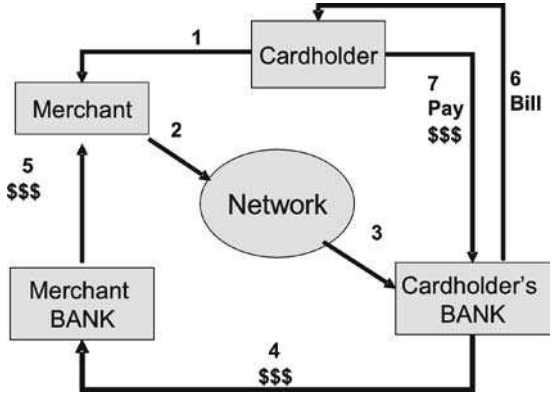


FIG. 2. Typical cardholder—merchant transaction cycle. Source: [85].

(1) *Contact.* Contact may be direct, proximate to, or in the vicinity of the reader. This depends on the type of card and the capability of the reader/terminal the cardholder wishes to use. In the case of a contact smart card, the card must be physically inserted into a reader/terminal and remain in contact with it during the session. In the case of contact smart cards, the reader/terminal verifies that the card is properly situated and that the power supplies of both card and

reader are compatible. It then supplies the card with the power necessary to complete the transaction through the chip's designated contact point (Cn). In the case of a contactless smart card, this requirement is eliminated.

(2) *Card validation.* The card must be validated in order to establish a session. Some reader/terminals have the ability to retain those cards found on a "hot list" of unauthorized cards and send a notice of attempted use to the card issuer. They may also be able to invalidate future use of a card. If the card is valid, however, the system captures an identification number to establish an audit trail and process the transaction using the authorized "value" for that transaction (for example, removing cash from an electronic "purse") that is stored on the chip. This activity is handled by a Secure Access Module (SAM) on a chip housed inside the reader/terminal. The SAM has the electronic keys necessary to establish communication with the defined operating system (OS) for the associated reader/terminal. Each OS is associated with a specified card platform, and the card must be compatible with that OS if it is to communicate with that particular reader/terminal and engage in a transaction.

(3) *Establishing communication*. A *Reset* command is then issued to establish communication between the card and the reader/terminal. Clock speed is established to control the session. In the case of both a contact and a contactless smart card, the reader/terminal obtains the required data from the card and initiates the requested transaction.

(4) *Transaction.* Data required for the session is exchanged between the card and the reader/terminal through the defined Input/Output contact point(s). A record of the transaction is stored on both card and reader.

(5) *Session termination*. For contact smart cards, the contacts are set to a stable level, the Supply Voltage is terminated and the card is then ejected from the terminal. For contactless cards, a termination protocol ends the session and resets the reader.

3. Smart Card Technology

3.1 Technology Overview

The growing interest in smart card technology is due to its ability to (1) withstand tampering and counterfeiting [59,73,97]; and (2) support multiple applications [21, 31,87] that are of growing interest due to the smart card's physical characteristics, data formats and information security capabilities. The smart card is a next generation "smart" identification card that offers the potential for tighter security and increased functionality, a step up from the earlier magnetic stripe card technology

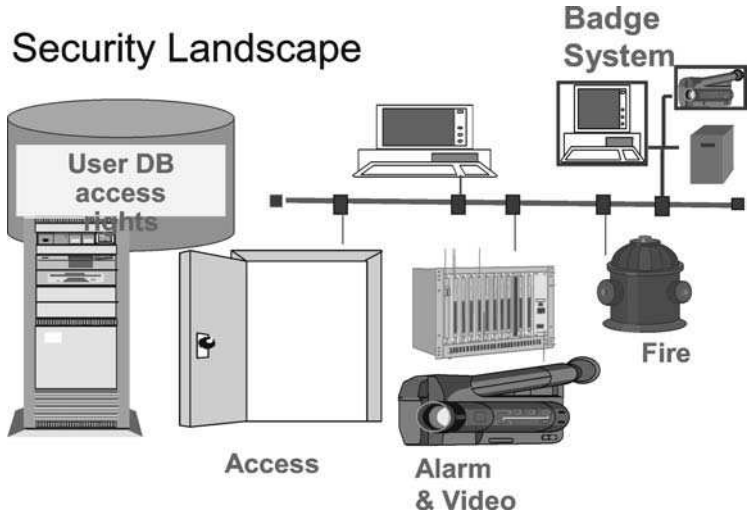


FIG. 3. Example of a physical access control system.

still used in most digital financial transactions at this time. This means that logical and physical access can be controlled from the same card (see Fig. 3).

Card transactions involve two or more parties (refer to Fig. 2 in Section 2 of this paper); for example, cardholder + cardholder's financial institution or [cardholder + cardholder's financial institution] ↔ [merchant + merchant's financial institution] [21]. The smart card is an improvement over simple magnetic stripe technology in that it is able to (1) store and *directly* distribute containered data that can represent more than one affiliated institution; (2) facilitate more than one type of authorized use; (3) carry substantially larger volumes of data on-board; and (4) process transactions at significantly higher rates of speed.

Today's smart cards may be either "contact" or "contactless," but all smart cards have on-board processing capabilities that require an integrated circuit (IC) chip that is physically located on the card. Applications may be split between cards and readers or off-loaded onto the card. Applets and small databases are now being designed to fit on smart cards.

3.2 Physical Characteristics

The physical size of the smart card is that of the magnetic stripe bank credit/debit cards in use today. This size is a compromise intended to promote backward compatibility with the existing infrastructure of financial transaction processing systems

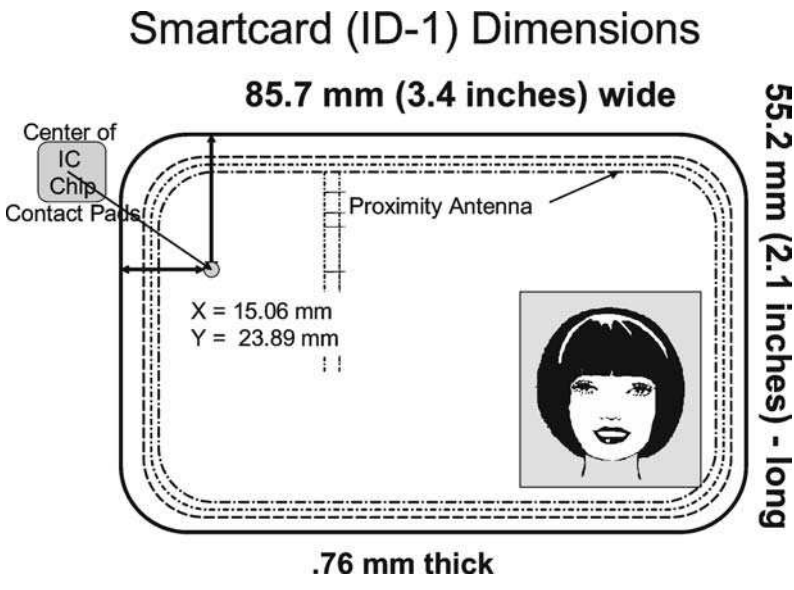


FIG. 4. Smart card physical dimensions.

that were developed prior to the advent of smartchip cards. Currently, international standards call for an "ID-1" plastic card base. Basic dimensions (not to scale) of a typical smart card are shown in Fig. 4.

The earliest smart cards were a laminated plastic sandwich. That is, two layers of plastic were bonded together. The bottom layer was solid and the top layer had a section cut out for the chip, which was then inserted and wired to the card. One example, a side cut-away, is shown in Fig. 5. This technology did not provide adequate physical protection from cardholder's "hip pocket" use and abuse. The chip tended to pop out and the card layers tended to peel and separate. As a result, the data on the card could not be read.

When a card could not be read, the eCash (stored value) on the chip could not be verified through an audit trail, so cardholders were victimized twice—first with a faulty card, and then with loss of the cash (stored value) recorded on the chip. This situation upset cardholders, the constituency of early adopters in the USA, as most were students, far from home and on restricted incomes.

In addition, cards and card applications were usually provided by a complex mix of third parties, so problems were significantly more complicated to resolve than had been the case with a 2-party identification card. As a result, card issuers, the customer base, were also unhappy. They had paid significantly more for this technology, yet it

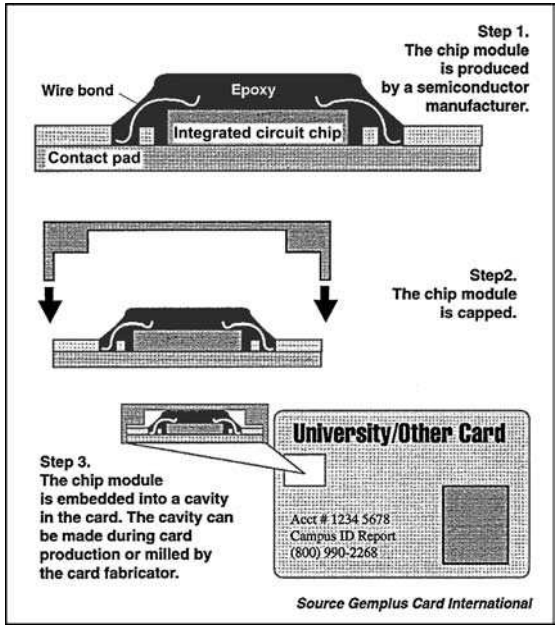


FIG. 5. Side view of an integrated circuit chip. Source: Gemplus; Campus ID Report.

damaged their relationship with their primary user populations. Solutions needed to be found, so manufacturers rapidly looked for ways to create a durable plastic card base.

One type of material, ABS plastic, or acrylonitrile-butadiene, was primarily used in injection molding. Another plastic, PVC, or polyvinylchloride, was also tried. It should be noted that there are several factors that impact the cost and quality of card printers: printing speed, duplexing (the ability to print both sides of the card in a single pass), encoding, and networking. Manufacturers also tried various printing technologies. They learned that one of the better ones is Dye Diffusion Thermal Transfer, or D2T2. In the D2T2 process, heat is used to transfer dye from a thin plastic "carrier" ribbon to the surface of the card. ABS plastic was soft and it did not work well with D2T2 printing technology. However, pure PVC was not much better. When pure PVC was used, layers separated and peeled and dyes did not adhere to the surface. Eventually, manufacturers found that they could increase the number of layers (to, for example, seven) and use thin layers of video grade PVC. To create video grade PVC, certain polymers/polyesters (polyethylene terephtelates: PETG, PETP, and PET) are added. This type of plastic works very well [22]. Manufacturers can also choose from a half dozen processes to cement these layers together.

Smart cards often contain a video image of the cardholder. Digital videography has replaced the tiresome processes of taking pictures, cutting them to fit, positioning them on a card and them laminating it all together. Images are captured using RGB (the best and costliest), S-VHS and NTSC video signals. Images are compressed and stored in formats such as JPEG. Image quality is determined by the size of the original image and the compression ratio. In the past decade, card production, products and processes have all improved. At the same time, costs have dropped dramatically. As a result, card quality is not the problem that it once was, but customers do need to become informed about their options. A number of publications, associations and conferences are now available.

3.3 Contact and Contactless Smart Cards

"True" smart cards may be either "contact" or "contactless" and the differences are explained below.

- *Contact cards* require direct contact between the card and the reader [31]. The smart card is placed into the reader. As a connection is made between the reader and the contact points located on the card's contact plate, an electrical circuit is completed [15]. Since contact smart cards do not have an on-board power supply, they must rely on the power supplied by the reader if they are to function. This is why contact smart cards must be inserted into a reader that makes direct contact with the chip. (See Section 4, Standards, for a discussion of the contact points of IC chips.)
- *Contactless* (*also called "proximity" or "prox"*) cards are designed to function without the need for the card to make physical contact with a card reader. Contactless smart cards do not use the electrical contact points found on the contact card's smartchip. Instead, they depend on electrical coupling [15]. The contactless card contains an on-board chip that enables the reader/terminal to (1) interact with the card over greater distances; (2) transmit larger volumes of data; and (3) transmit data at faster rates of speed. Although other forms of contactless access control technologies do predate the contactless smart card, the impetus for the development of a contactless smart card, introduced in 1988, was the perceived need to accelerate the speeds at which physical access transactions were processed and cardholder access approved and recorded.

3.4 Physical Characteristics of the Integrated Circuit (IC) Chip

The smart card, unlike the simple magnetic memory card, contains an on-board electronic module ('processor') and one or more silicon-based IC chips. The IC chips

used on a smart card are primarily based on (Motorola) semiconductor technology [33]. These chips contain both short- and long-term memory cells [31] ranging from a few hundred bytes to several megabytes of random access memory (RAM). RAM provides a "scratch pad" for processing information. The card controller software is stored in permanent nonvolatile read-only memory (ROM). Today's smart cards may also have special circuitry related to encryption.

There are three varieties of integrated circuit (IC) cards: (1) serial memory, (2) hardwired, or wired, and (3) the 'true' smart card, that contains a microprocessor.

- *Serial memory* (simple memory; synchronous) cards provide 1–5 Kb of storage but do not have processing capability. The memory card stores data in non-volatile memory and the card contains only sufficient logic circuitry to manage control and security functions [42,31]. The transaction is simple: a write function simply transfers the electronic equivalent of cash from the card to the reader (think of it as a digital "cash register"). The value transferred to the reader is then transferred to a traditional bank account [31].
- *Protected memory* (also called "hardwired" or "wire") cards have two sections of memory: (1) a read-only (RO) portion that is programmed as part of the manufacturing process; and (2) a read-write (RW) area that is reserved for software applications. The use of the terms "hardwired" and "wired" are derived from the wired circuitry that characterizes the memory allocation in the chip (not the wires that physically link the chip to the card or the antennae in contactless cards).
- *Microprocessor* cards are the 'true' smart cards because they have on-board processing capabilities. This provides an additional layer of security, because the encryption operations can be conducted by the card itself without reliance on external hardware to determine encryption keys and security.

A cut-away (side, top to bottom) view of the typical stacking order of the microprocessor-enabled smart card [51] is shown in Fig. 6. This diagram represents

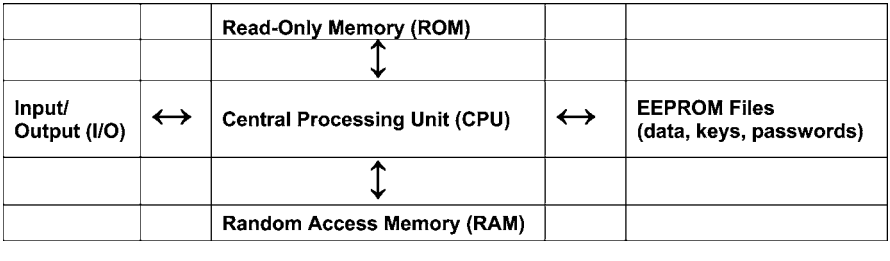


FIG. 6. Architecture of a smart card electronic module. Source: [51].

the 'top' layer, the electrical components, of an embedded smart card processor chip. Some chips may not include all possible memory types, and additional nonvolatile memory type NVM is not represented. Security is increased and card size is minimized through the combining of all of the depicted elements into one integrated chip [51].

In the case of contact cards, the IC chip must make physical, electrical contact with a card reader in order to complete the specific transaction(s) for which the chip is programmed. This is because the chip's source of electrical power is in the reader and not on the card. The contactless smart card need not make direct contact with the card reader, however, because it emits low frequency radio waves that interact with the reader at varying distances.

3.5 Processor Capacity

Smart cards include an embedded silicon-based processor, and they can also include a cryptographic chip for data encryption [8]. The first (8-bit) processors were almost as powerful as the desktop personal computers of the 1980s [34].

Processors have since evolved through 32-bit on-board processors [42] to the still rare 64 KB CPU. The three primary constraints on processor size today are (1) market acceptance based on pricing constraints; (2) lack of useful applications available to card issuers; and (3) EEPROM memory (http://www.cyberflex.com/Products/Cyberflex_Access/cyberflex_access.html).

There are three types of card-based contactless technologies: (1) close coupling (Type A and Type B); (2) proximity cards; and (3) vicinity cards as well as several nonstandard proprietary systems. Most contactless cards in use today are close coupling, Type A. These do not use a microprocessor. However, Type B close coupling contactless cards do use microprocessors and new technologies are in development that will eventually require them. These three types are discussed in more detail in the section of this paper that addresses smart card standards.

The earliest contactless smart cards were "read only." This means that the reader does not send any data to the card. The next step up, the memory cards, were designed to allow readers to both receive and send data (read-write) to the card. This means that cards could now be updated with each transaction. Hybrid cards (those with both contact and contactless IDC chips) use separate chips for these functions, as contactless cards are designed differently. On some cards, where these chips are actually connected, the contact interface drives both the contact interface and the contactless interface. Figure 7 represents one of the more sophisticated forms of contactless smart card designs available today.

Regardless of the actual technology used, the dominant characteristic of all contactless cards is that they can be activated and used at a distance. Cards and card

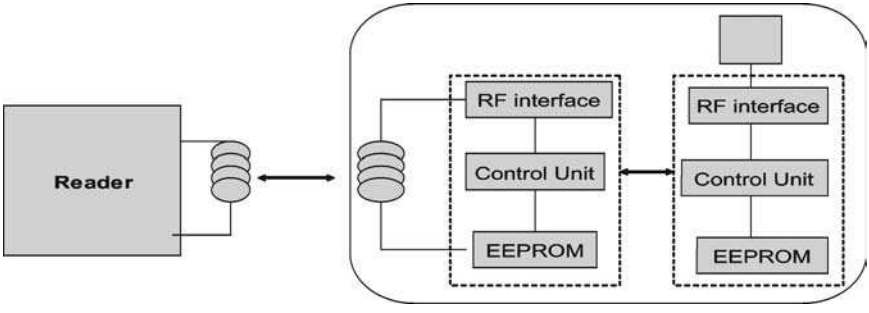


FIG. 7. Hybrid contactless cards with connected chips. Source: [65].

readers/terminals transmit radio frequency (RF) emissions that support electrical coupling between cards and associated readers [73]. The card reader supplies the card with the power necessary to run the card's microprocessor [34].

3.6 Current Specifications

According to the corporate web site for Axalto (http://www.axalto.com and http://www.gemplus.com), typical current smart card technical and security specifications include the following:

- *Technical*. Multi-application capable (4 Kb to 64 Kb and larger) EEPROM, capable of withstanding 700,000 cycles; capable of retaining data for up to ten years; an 8-bit CPU microcontroller; global personal identity number (PIN) capability that includes PIN sharing by applications; interoperable and compliant with international standards, including ISO 7816; (Sun) Java Card 2.1.1 and Open Platform 2.0.1; external clock frequency: 1 to 7.5 MHz; sleep mode (locks down the card to prevent unauthorized use); and capable of operating in temperatures that range from -25 to $75\,^{\circ}$C.
- *Security*. Extended support for a variety of encryption and digital signature standards: DES, T-DES, RSA, SHA-1; fire walled applets; multi-stage update/load/install register for enhanced security; and secure channels for application dynamic loading and deletion, data update and card life cycle management. Support for long integer modulo arithmetic on RISC-based smart cards is currently the subject of study as the modulus should have a length of at least 1024 bits, but this is difficult to embed on a card due to software constraints and clock speed limitations [39,35,52,89].

Recent innovations in contactless smart cards appear to target increasing the on-board memory to absorb additional applications [7,10] and the mini databases being designed to fit on smart cards. One recent consideration with promise is the notion of mobile cookies [16] that allow cardholders and/or card issuers portable, personal access histories that are independent of the computer from which the access was requested.

4. Smart Card Standards

4.1 Smart Card Standards Organizations

The global nature of digital transaction processing, particularly for financial transactions, mandates some agreement on standards for cards and readers. There are two organizations involved in the development and issuance of international smart card standards: the American National Standards Institution (ANSI) and the International Organization for Standardization (ISO) [2,18].

- *ANSI*. In the US, the appropriate organizations are the American National Standards Institution (ANSI), the National Committee for Information Technology Standards (NCITS), the Cards and Personal Identification Committee (B10) and its Contactless Technology Subcommittee (B10.5). Publications from this group are coded ANSI/NCITS/B10/B10.5.
- *ISO*. The International Organization for Standardization (ISO) (http://www.iso.ch/), representing some 130-member nations, establishes standards that facilitate international manufacturing, commerce, and communication. ISO determines the international standards for the physical characteristics and data formats of smart cards. The International Electro Technical Committee (IEC) focuses on electro-technical standards that facilitate the growth of international manufacturing and support services and the Joint Technical Committee (JCT1) develops standards for information technology. ISO Subcommittee 17 focuses on identification technologies, including ID cards and ancillary technologies and its working group, ISO Working Group 8 (WG8), is the level at which international standards for contactless smart cards, the latest evolution in smart cards, are developed (ISO/IECJTC1/SC17/WG8) [42,3].

For a list of the most relevant smart card standards to date, see Table IIIA: ISO/IEC Standards and Table IIIB: Contactless Cards. Additional information on magnetic stripe cards is found in Section 5, Associated Access Technologies. Standards are available from ISO (http://www.iso.org). ANSI (http://www.ansi.org) and others all track and report on standards —e.g.,

TABLE IIIA
ISO/IEC STANDARDS[a] (DATE AS OF LATEST FULL/PARTIAL REVISION)

Date	Standard	Content
2003	7810	Physical characteristics smart card (ID-1 size)
1997	7811	Magnetic stripe data formats
2002	7811-1	Embossing
2001	7811-2	Magnetic stripe—low coercivity
1995	7811-3	Location of embossed characters on ID-1 cards
1995	7811-4	Location of read-only magnetic Tracks I and II
1995	7811-5	Location of read-write magnetic Track III
2001	7811-6	Magnetic stripe—high coercivity
2001	7813	Financial transaction cards
1998[b]	7816-1	Physical characteristics
1999	7816-2	Location, size of the IC card's electronic contacts
1997	7816-3	Electrical signals
1995	7816-4	Defines, in part, the structure of stored files
1994	7816-5	High-level application communication protocols

[a]ISO 4909:1987 sets forth the standard for data content for this track. Two related standards, not discussed in this paper, address optical memory (ISO/IEC 11693 and ISO 11694).

[b]Amended 2003.

TABLE IIIB
CONTACTLESS CARDS

Latest date	Standard	Content
2001	10373	Test Methods—Proximity Cards (Part 6) Test Methods—Vicinity Cards (Part 7)
2000	10536-1	Close Coupled Cards—Physical Characteristics
2001	10536-2	Close Coupled Cards—Location and Size of Coupled Areas
1996	10536-3	Close Coupled Cards—Electronic Signals, Reset Procedures
2000	14443-1	Proximity Cards—Physical Characteristics
2001	14443-2	Proximity Cards—Radio Frequency (RF) Power, Signal Interface
2001	14443-3	Proximity Cards—Initialization and Anticollision
2001	14443-4	Proximity Cards—Transmission Protocols
2000	15693-1	Vicinity Cards—Physical Characteristics
2000	15693-2	Vicinity Cards—Air Interface and Initialization
2001	15693-3	Vicinity Cards—Anticollision and Transmission Protocols

Source: http://www.iso.ch/iso/en/ISOOnline.frontpage; http://public.ansi.org/; and http://www.cyberd.co.uk/support/technotes/isocards.htm (among others).

- http://www.cyberd.co.uk/support/technotes/isocards.htm
- http://www.incits.org/scopes/590_1.htm

- http://www.blackmarket-press.net/info/plastic/magstripe/Magstripe_Index.htm
- http://www.cardtest.com/specs.html
- http://www.javacard.org/others/sc_spec.htm

4.2 Early Smart Card Standards

Three standards specifically address the physical and electronic formats of magnetic strip cards: ISO 7810 (especially parts 2 and 6), ISO 7811, and ISO 7813.

ISO 7810. The earliest international standard for IC cards, this standard established the physical location of the chip on the card (see Fig. 4, above). As previously explained, the stipulation that dictated the chip's actual physical placement on the plastic card base was a result of the demand by financial institutions for maximum backward compatibility with existing magnetic stripe systems, as well as the desire to provide maximum protection for the chip (i.e., if the card was bent, for example) [33].

ISO 7811. This standard specifies card data formats (for example, Farrington 7B as the specified font). Specific details in the standards address embossing, as well as the location and data formats of the two read-only tracks (I and II) and the read-write track (III). For more on magnetic stripes, see Section 5 of this paper.

ISO 7813 sets forth the specifics of financial transaction cards.

4.3 Contact Smart Card Standards

One standard (ISO/IEC 7816) covers integrated circuit cards with contacts (contact smart cards).

ISO 7816. The various sections of this standard describe physical characteristics of the smart card. Figure 8 is an example of the contact configuration of an IC chip (refer to Table IV). Part 1 covers the physical characteristics of the smart card. Part 3 specifies electrical signals. Part 4 defines, in part, the structure of stored files. Part 5 covers High-level application communication protocols. Part 2, which covers electrical contacts, is described below.

- *Part 2.* Location and size of the electronic contacts on the smart card. This standard specifies six (6) contact points, although some chips have more. Each contact (designated C*n* below) has its own defined function:

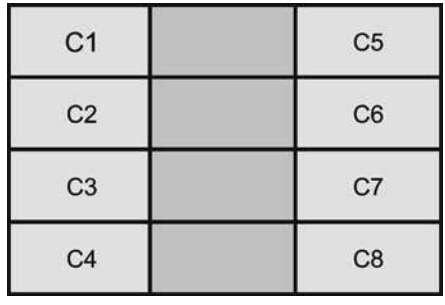


FIG. 8. IC Chip contacts.

- C1 provides the necessary power—supply voltage (VCC) current (5V ± .5V DC)—required by contact smart cards, as they do not have any power of their own.
- C2 is the Reset (RST) that establishes communication.
- C3 is the Clock (CLK) that sets the rate of data exchange.
- C4 is not used and need not be present on the chip.
- C5 is the Ground (GND).
- C6 is not used and must be electronically isolated on the chip.
- C7 is the Input/Output (IO). This contact is used to transmit data from the terminal to the chip or to receive data by the terminal from the chip.
- C8 is not used and need not be present on the chip.

4.4 Contactless Smart Cards Standards

There are three primary standards for contactless smart cards: (1) ISO/IEC 10536, (2) ISO/IEC 14443 Proximity Cards, and (3) ISO 15693 Vicinity Cards. A comparison of their Communications Parameters is shown in Table IV, located at the end of this section).

ISO/IEC 10536 Close Coupling Cards. The first standard for contactless cards required that cards either be inserted into a reader or placed in a very precise location on the card reader's surface. The limited distance and the high level of accuracy required for a "good read" discouraged the use of contactless cards in controlled environments. This standard has been abandoned as a result of improvements in contactless card technologies.

TABLE IV
A COMPARISON OF PARAMETER VALUES FOR CONTACTLESS CARDS

Standard: 14443 Proximity Cards				
Type	Type A (memory; MIFARE®)		Type B (microprocessor)	
Reader to Card	Frequency	13.56 MHz	Frequency	13.56 MHz
	Modulation	100% ASK	Modulation	10% ASK
	Bit coding	Modified miller	Bit coding	NRZ
	Data rate	106 kb/s	Data rate	106 kb/s
Card to Reader	Modulation	Load	Modulation	Load
	Bit coding	00K	Bit coding	BPSK
	Subcarrier	847 kHz	Subcarrier	847 kHz
	Bit coding	Manchester	Bit coding	NRZ
	Data rate	106 kb/s	Data rate	106 kb/s

Standard: 15693 Vicinity Cards		
Card to Reader	Frequency	13.56 MHz
	Modulation	100% ASK or 10% ASK
	Data coding	1 of 4 or 1 of 256
	Data rate	1.65–26.48 kb/s
Reader to Card	Modulation	Load
	Bit coding	Manchester
	Subcarriers (1 or 2)	Fs1 = Fc/32 = 423.75 Mz; Fs2 = Fc28 = 484.28 MHz
	Bit coding	Manchester
	Data rate	Fs1 = 6.62–26.48 kb/s; Fs2 = 6.67–26.69 kb/s

Proprietary interfaces				
System	Cubic		Sony	
Card to Reader	Frequency	13.56 MHz	Frequency	13.56 MHz
	Modulation	5%–8% ASK	Modulation	10% ASK
	Bit coding	NRZ	Bit coding	Manchester
	Data rate	1l5.8 kb/s	Data rate	106 or 212 kb/s
Reader to Card	Subcarrier	No	Subcarrier	No
	Modulation	ASK Load	Modulation	ASK Load
	Bit coding	NRZ	Bit coding	Manchester
	Data rate	115.8 kb/s	Data rate	106 or 212 kb/s

Source: [3].

ISO/IEC 14443 Proximity Cards. The large majority of proximity cards adhere to this standard. ISO 14443-2 has two types of interfaces, referred to as Type A (memory cards, also called MIFARE® cards) and Type B (microprocessor cards). Type A cards are often called MIFARE® cards because the standard, developed beginning in 1994, includes the MIFARE® technology patented by Micron, an Austrian company that was purchased by Philips. Micron's technology was, as a condition of inclusion in the standard, made available to others via licensing. Type B began as a higher-security microprocessor (only) card, but expanded to include both memory and cryptography. Type B card chips have been mostly provided by STM, lnfineon. Samsung, and Atmel. Initially, Type A and Type B technologies were complimentary, but with the entry of competing technologies, they do currently compete. Even so, Type A cards account for roughly half of all contactless smart cards issued, according to the Philips corporate web site (http://www.semiconductors.philips.com/news/content/content/file_798.html as of February 2002). The remaining card base is mostly Type B cards plus two proprietary systems: Cubic (GO-Card®) and Sony (FeliCa®). The useful range of connectivity, which is the "read range" for smart cards that adhere to ISO standard 14443, is up to 10 cm, depending on power requirements, memory size, CPU and co-processor. There are four parts to this standard:

- *Part 1. Physical Characteristics—International Standard IS 4:2000.* Card size is set at 85.6 mm × 54.0 mm × .76 mm. This is the same size as a hank credit card, so it continues to support backward compatibility with the large number of magnetic stripe systems that support most transaction processes in use at the present time.
- *Part 2. Radio Frequency Power and Signal Interface—IS 7:2001*. Both Type A and Type B interfaces operate at the same frequency and use the same data transmission rate. They differ only in modulation and bit coding.
- *Part 3. Initialization and Anticollision—IS 2:2001*. This sets the communication protocols that establish the initial connection between the card and the reader when the card enters the reader's radio frequency (RF) field. Anticollision addresses situations where more than one card enters the targeted magnetic field at the same time. This part of the standard system determines which card to use in the transaction and ensuring that all cards presented are inventoried and processed.
- *Part 4. Transmission Protocols—IS 2:2001*. Transmission protocols define the data format and data elements that enable communication during a transaction.

ISO/IEC 15693 Vicinity Cards. The vicinity card has three modes with associated ranges of operation: (1) read mode (70 cm); (2) authenticate mode (50 cm); and (3) write mode (35 cm). There are three separate parts to this standard:

- *Part 1. Physical Characteristics—IS 7:2004* establishes the physical card size at the ID-I size (145.6 mm × 54.0 mm × .76 mm). This is the same size as a bank credit card.
- *Part 2. Air Interface and Initialization—IS 11:2001* defines frequency modulation, data coding and data rate values for both reader-to-card and card-to-reader communication.
- *Part 3. Anticollision and Transmission Protocol—IS 6:2001* defines the protocol, command set and other parameters required to initialize communication between a reader and a card. It also defines the anticollision parameters, which facilitate the selection and appropriate use of a single card when multiple cards enter the reader's magnetic field.

4.5 Nonstandard Contactless Technologies

There are also proprietary contactless interfaces, mainly variations on ISO 14443, that use nonstandard bit rates and/or bit encoding methods and lack a subcarrier. Two examples are Cubic Corporation's GO-Card (US and England) and Sony's FeliCa card (Hong Kong and several Asian countries). Cubic and Sony tried and failed to obtain standards classification for these products under ISO 14443.

4.6 Comparison of ISO/IEC 14443 and ISO/IEC 15693

Contactless cards differ in several ways, among them transaction speeds, anticollision techniques and data security:

- *Transaction speeds.* The rate and volume of data exchange involves four events: (1) Input/Output (I/O), (2) memory access, (3) encryption and (4) processing. Together, these activities comprise nearly one-quarter of the total time used in a single session. Sophisticated, highly secured transactions benefit from adherence to the ISO 14443 standard, as ISO 15693 smart cards are much slower when handling a large volume of data. Where the read range is important, it should be noted that ISO 14443 microprocessor cards have a shorter read range than ISO 14443 memory cards and ISO 15693 cards have greater distance. Originally developed for such functions as inventory control, these cards are considered less secure for physical access control—the greater the distances involved, the greater the risk of unintended access authorization. Data collisions are also more likely at greater distances and slower transaction processing speeds.

- *Anticollision techniques.* When multiple cards enter a reader's range, anticollision techniques are used. Most ISO 14443 smart cards (both Type A and Type B) use the *Bit Collision* technique, where the reader sequentially inventories cards and establishes working sessions. Another option is the *Time Slot* technique, where cards are polled by the reader after a collision and assigned time-based response times. ISO 15693 uses the *Slot Marker* anticollision technique, where the reader inventories each card in its read range and assigns a specified slot for the card's response so that it can locate and work with each card in its read range. Any of these techniques is sufficient for physical access control.
- *Data security.* Authentication and encryption are performed with a key-based cryptographic function. The keys are generated by a random number generator that is designed as part of the IC. The key (or number) is then incorporated into an algorithm residing on the 1C. The algorithm function is computationally intensive and should be supported by a dedicated hardware coprocessor. The circuitry involved is specially designed to perform the complex computation and makes using the supported cryptography viable without affecting transaction time and power consumption. Common examples include Data Encryption Standard (DES), the Digital Signature Algorithm (DSA) and RSA (named for Rivest, Shaman and Adleman who developed it in 1977); DSA is only used for signatures, not encryption. Elliptical Curve Cryptography (ECC) has become a popular method and is supported by a number of 1C providers with dedicated cryptographic engines housed on the chip [3, 62].
- *Public key*, or symmetric key, cryptography creates uses a single public key that is traded between sender and recipient, usually over open lines. DES is a symmetric, or public key, encryption system. For example (see Fig. 9):

  (1) Jean uses the public key directory to encrypt his public key and creates a *hash*, or document digest. The information used to create Jean's hash generally includes version#; serial#; signature algorithm; card issuer's name; card expiration date; subject name; the cardholder's public key; card issuer unique id; cardholder's unique id; and extensions [29]. Both Jean and the Certificate Authority (CA) gets (decrypted) copy.

  (2) Second, Sue acquires Jean's Public Key from the CA and uses it to decrypt both his document digest and his document. This lets Sue assume that Jean is really Jean (and not John who is pretending to be Jean).

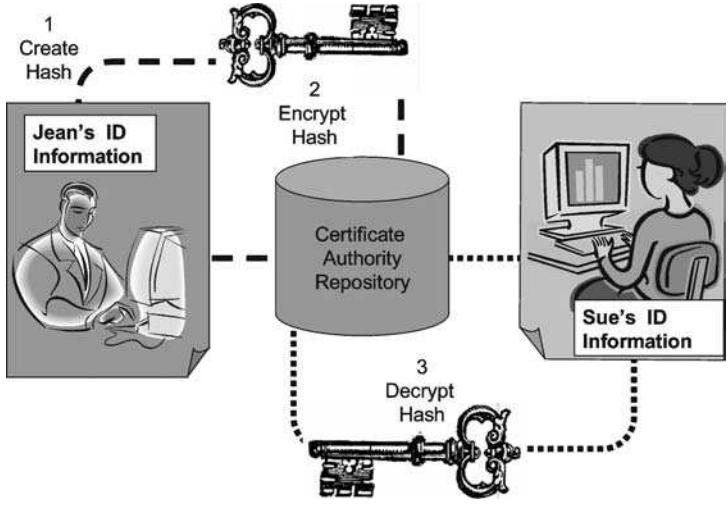


FIG. 9. Data Encryption Standard (DES): A public key symmetric encryption system.

(3) Sue then encrypts and sends a response to Jean, using Jean's public key. This also creates a "hash" or digest. Jean is able to decrypt and read the response document from Sue. If Jean's hash matches Sue's hash, then Jean can assume that Sue is really Sue (and not Sarah pretending to be Sue).

- *Type 2. Private-key, hidden key* or *asymmetric-key* cryptography is a scheme in which each user has both a public key and a private key. The public key is distributed to others while the private key remains a secret. One key is used for encryption and the other for decryption. RSA is one example of this type of encryption system.
- By *card cloning* and *counterfeiting*, we mean the unauthorized recording of a transaction/session that is used to forge a real transaction. Smart cards address these problems with a process called dynamic authentication, in which the challenge and response between reader and card change with each transaction. A random number generator must be used for security purposes, as it generates unique keys for each session. If nonrandom numbers are used, the encryption scheme develops a pattern. If this pattern is detected, the encryption codes can be much more easily broken.

Nonrepudiation is achieved through cryptographic methods that prevent an individual or entity from denying previous participation in a particular transaction. The

fact that a third party can verify the transaction prevents repudiation of the transaction by either of the two parties that participated.

4.7 The Role of Standards

Over the past 25 years, there has been an explosive growth of international financial transaction processing accompanied by an equally explosive growth in economic (and associated violent) crimes. Smart cards could contribute significantly to reductions in counterfeiting and tampering that are associated with financial fraud, but they must meet the needs of financial institutions that still have an enormous investment in the older magnetic stripe technology. Future innovations include a growing number of truly global applications. Approval and adoption of new standards is sometimes about first mover advantage. For example, Micron's patented Mifare® technology was incorporated, but Sony's FeliCa® card interface, after much study, was not. New standards supported by companies and industry associations revolve around a Common Access Card (CAC) that would support a wide range of devices and applications with global reach. The smart card has its limitations, and the growing number of interested parties is moving to address this issue. (See, for example, www.globalplatform.org, java.sun.com/products/javacard/specs.html, and http://www.sun.com/aboutsun/media/presskits/iforce/ActivCard_SP_final.pdf).

Problems do arise, of course. First, governments assign radio frequencies (1) to prevent conflicts within their borders; and (2) to try to impose their standards on others. The radio frequencies for contactless cards in the US are not compatible with those of other nations. It is likely that international standards will apply. Given that the read ranges are so small, this conflict should not interfere with any *domestic* system that might make use of the same radio frequency, but this lack of compatibility does exist and it should be noted. Second, some locations do not have this capability while others have run out of it.

5. Associated Access Technologies

The smart card often includes two other technologies: (1) a magnetic stripe that facilitates backward compatibility with financial (and other) transactions; and (2) the barcode, that facilitates contactless access for purposes such as inventory control. These two technologies are still widely used in a range of settings, they are relatively inexpensive and they are usually less complex to administer than smart cards. They are often incorporated into applications on the smart card itself. For this reason, a brief discussion of two of these associated access technologies is in order.

5.1 Electro-Technology Access: The Magnetic Stripe

The cardholder's personal data is stored on the smart card in electronically erasable programmable read-only memory (EEPROM) that is an "alterable non-volatile memory" [31]. Data storage on smart cards varies, depending on the requirements of the application. However, it can all be traced back to the mid-1980s, when the ISO subcommittee on information processing systems, known as Technical Committee 97 (TC 97), issued ISO 7812, Identification cards: Numbering system and registration procedure for issuer identifiers. This standard described a standardized card numbering scheme for transaction routing and control. To ensure that card readers could find and interpret the number, a magnetic stripe was needed to hold what came to be known as an "ISO number." The American Bankers Association (ABA) magnetic stripe-encoding standard is used on magnetic stripe cards today, which are discussed below [53,54,94].

The ISO number was a key success factor for smart cards. It encouraged the invention of card readers that could locate and correctly interpret data supplied by multiple card issuers. This, in turn, led to the development of cost-effective point of sale (POS) and ATM networks (see Fig. 10 for an example of an ISO number) [94].

- *Major Industry Identifier* (*MII*). The first two digits, the MII, indicate the number of digits in the Issuer Identifier that immediately follows it. Most cards use 3, 4, 5, 6, or 7 as the MII. With limited exceptions, the Issuer Identifier for all these MIIs has five prescribed digits following the MII (for a total of six digits). Each card starts with the same six digits, except for giants and "wastrels" who run out of numbers and need more. This number indicates the category best describing the card issuer:
  1 = airlines
  3 = travel and entertainment
  4 = banking/financial
  5 = banking/financial

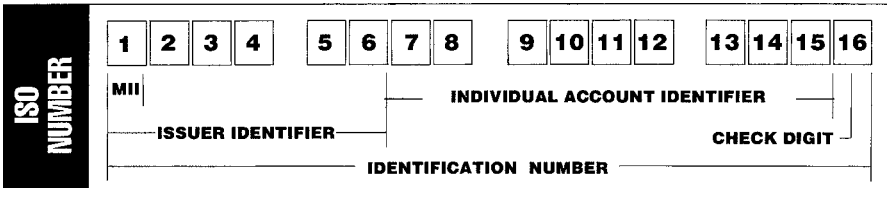


FIG. 10. ISO number.

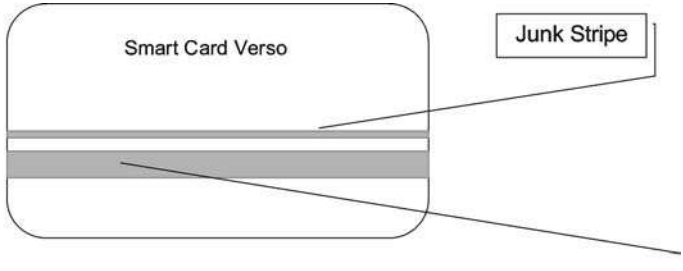
6 = merchandizing and banking

7 = petroleum

- *Individual Account Identifier*. The next nine digits are the unique identifier of the individual to whom this number has been assigned. In the case of smart cards, they 'name' the cardholder. Issuers may generate these numbers using any logic they choose. In order to prevent fraud, most issuers use a "skip" to avoid having sequential valid numbers. The final digit is a check digit, calculated by applying a simple mathematical formula to the preceding 15 digits. This check allows a card reader to perform the calculation based on the first 15 digits it read from the "mag" stripe and check it against the final digit read. If they match, it is likely a valid read [4].

The data track standard, formalized in ISO 7811, determines the composition and location of the primary magnetic stripe (sometimes called the *ABA stripe*) as shown in Fig. 11. This stripe actually consists of three separate tracks, each 0.110 inches in height:

- *Track 1* supports 79 read-only alphanumeric characters with a compression rate of 210 bits per inch. It was initially designed by the International Air Transportation Association (IATA) to support passenger ticketing and reservation data.



Track	Developer	Capacity	Compression
1	IATA	79 alpha/numeric	210 BPI
2	ABA	40 numeric	75 BPI
3	Thrift	107 numeric	210 BPI

FIG. 11. Magnetic stripe characteristics.

- *Track 2* supports 40 read-only numeric characters that are compressed at 75 bits per inch. It was developed by the American Banking Association (ABA) to support financial transaction processing, so the first field is typically the cardholder's account number.
- *Track 3* was designed by the Thrift Industry to support a read-write function that enables this track to be updated with each transaction.

It should be noted that some smart cards also include what is commonly called a *junk stripe.* This stripe primarily serves as an anonymous debit (declining balance) feature where personal identifiers are not required. For this reason, its use is generally restricted to proprietary closed systems, primarily those involving cash-based vending, i.e., copiers, laundromats, city transit, telephones, soda machines and snack machines. This stripe may be located on the front or back of the smart card. If located on the back, it is typically positioned slightly above the 3-track magnet stripe described above.

Utilizing all three tracks, an ABA-compliant magnetic stripe card has a total capacity that is limited to 226 characters. It should be noted that even the smallest smart card chips have considerably larger capacity, but the smart card advantage rests in improved data security and additional functionalities, not in relative data storage capability.

5.1.1 Coding the Magnetic Stripe

As is true for all magnetic recordings, microscopic ferromagnetic particles, just millionths of an inch in length, are containered in a resin-based coating that covers the location of the track. These tiny particles hold their most recent magnetic polarity when acted upon by an external magnet field. On an un-encoded stripe, all particles align in the same direction. The coding process uses magnetic "coercivity" to realign the magnetic polarity of selected particles located in certain positions on the coated track. Since only two polar directions exist (north/south, south/north), binary calculations can be used to record these states as zeros and ones. Tracks may be high coercivity (HiCo) or low coercivity (LoCo). This characteristic is determined by the amount of magnetic pull (measured in oersteds) that is required to flip the polarity of the target particles. HiCo stripes (coded at 1000 oersteds) are typically used to organize data in fields. LoCo stripes are encoded at 300–1000 oersteds. While LoCo cards are less costly to produce and encode, they are more susceptible to demagnetization and can easily be damaged in certain environments (intensive care units of hospitals, for example).

Two binary data calculation formats that are used to encode data on the magnetic stripe are the 5-bit ANSI/ISO *BCD Data Format* and the 7-bit ANSI/ISO *Alpha Data Format.*

- *BCD Data format*. 4-bits of data (5-zeros and ones) are used to create a 16-character set ($2 \times 2 \times 2 \times 2 = 16$) [12] that consists of the ten numeric digits (0–9), 3 digits for framing and 3 digits for control. The fifth bit is treated as a check device.
- *ANSI/ISO Alpha Data Format.* This 7-bit data format generates a sixty-four character set (ten numeric characters, all 26 letters of the alphabet, 3 framing characters and 18 control characters) using 6-bits for character generation and 1-bit as a check.

In both formats, there are at least 3 control characters. The *Start Sentinel* (SS) signals the start of meaningful data. This gives the card reader a chance to synchronize and decode the transmitted data. The *End Sentinel* (ES) control character is followed by the *Longitudinal Redundancy Check* (LRC) character that works as an error check for the whole line of data.

5.1.2 Barcodes

In 1948, Dr. Joseph Woodland, then a lecturer in mechanical engineering at the Drexel Institute of Technology (now Drexel University), became interested in the need for supermarkets to track inventory and automate the checkout process. He found that the variance in polychromatic systems was too great, but Morse code lacked enough elements to support the necessary level of detail. By extending those dots and dashes to create thin and thick lines, he and Bernard Silver, developed a system to decode the lines that called for the early equivalent of a laser light. They received US patent 2,612,994 in 1952 for this "Classifying Apparatus and Method." In 1973, Woodland's invention became the basis of the Universal Product Code, or UPC, an example of which is shown in Fig. 12 (see also Fig. 13).

Today, barcodes are assigned to products and used to link products to inventory and sales management systems in every sector of the economy. According to data compiled by the Uniform Code Council, UPC codes serve over 600,000 manufacturing companies and are scanned 5 billion times a day, but this is less than half of today's bar code technology. In libraries, for example, barcodes are generated and used as unique identifiers for both individual library patrons and individual items such as circulating books. ISO numbers (discussed in Section 5.1) are converted into barcodes ($2 \times 2 \times 2 \times 2 = 16$). Today, mini-databases of various types are being designed to be carried on the smart card itself. ISO numbers and UPC codes are examples of the types of data stored in these databases (http://www.uc-council.org and http://www.drexel.edu/coe/news/pubs/coepuzzleranswerinsummer2003.html).

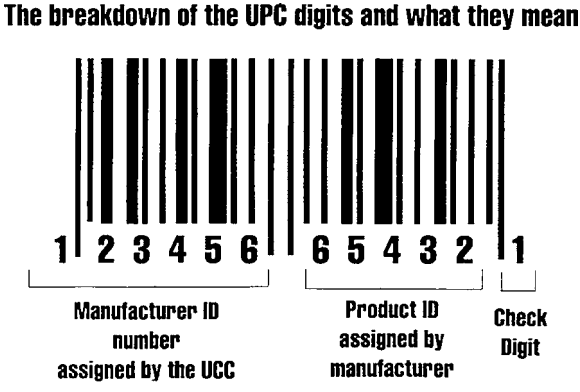


FIG. 12. Uniform product classification codes.

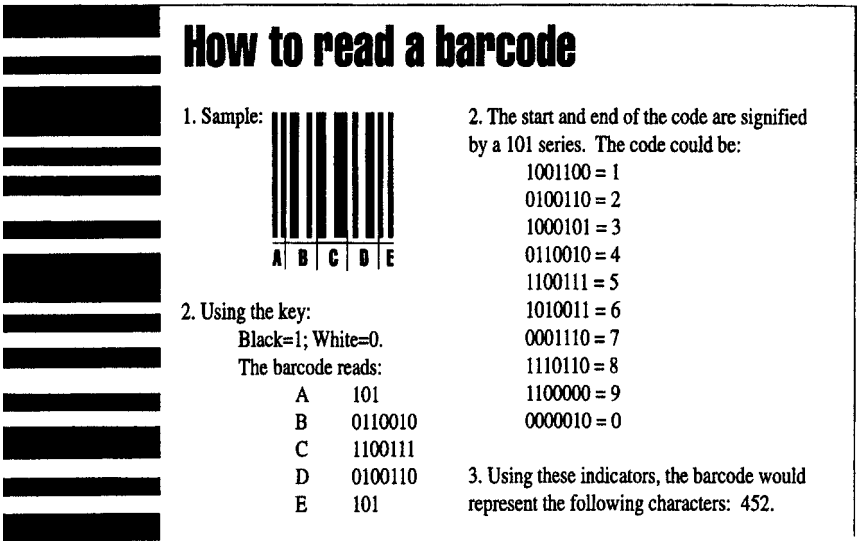


FIG. 13. How to read a bar code. Source: [4,5].

5.2 RISC-Based Smart Cards and The Java Virtual Machine

The earliest Application Data Protocol Units, or ADPU (addressed in ISO/IEC 7816) were developed and used to transfer messages between clients and servers (in this case, applications/processes split between cards and readers) [16]. Prior to the

development of the Java card platform, ADPU were awkward, idiosyncratic and took a long time to write. This made smart card applications proprietary, expensive, and very slow to develop. The stated purpose of Sun Microsystem's Java card platform is to create an open programming architecture so that applications can be written once to run on all cards. This would make it possible to program a smart card application "in a day." To accomplish this goal, Sun posts documentation and provides training classes that include constructing and parsing ADPU (see http://java.sun.com).

5.3 Multiple Applications

Today, the smart card is able to support multiple functions. Personal, portable biometrics are an area of growing interest. Examples of biometrics technologies that are being placed on the card include digital signatures, photographs, fingerprints, retina and iris scans and special forms of passwords. Some applications share PINs. Examples of applications that are currently in use include:

- Logical security—email, login, disk encryption, remote access.
- Physical security—facilities, equipment lockers, parking services.
- Electronic security—financial services (eCash, eBanking), employee benefits (401 K retirement plans, and healthcare).

6. Smart Card Security

6.1 Physical Security

There are several ways in which smart cards can be physically damaged and/or destroyed: (1) cardholders, (2) the environment, and (3) tampering.

- *Cardholders.* Individual cardholders do not take special precautions with smart cards, treating them much as they do cash, which can be bent, folded, stapled and mutilated without losing its value. Cardholders can, and do, bend cards, scratch chips, and break electrical contacts. Educating cardholders is not the answer. Cardholders should be offered additional—not replacement—functionalities. Cards should not be harder to maintain than cash. Smart card technology that does not meet this condition is at risk of substitution and/or abandonment. Newer manufacturing technologies and the creative use of protective sleeves (printed with sports schedules, emergency contact numbers, etc.) are the best defense against cardholder abuse.

- *Environment.* A range of environmental conditions can damage and/or destroy cards. Examples include voltage, frequency of use, humidity, light and temperature. Not every threat materializes, of course. For example, eelskin wallets do not destroy the magnetic data on the card; this is an urban myth that developed because eelskin wallets became popular at the time that smart card technology was new and often faulty [26]. However, the PVC base of the card does react to extremes in temperature. Data is certainly damaged and destroyed in some settings, such as hospital intensive care wards, where a lot of electrical equipment is in constant use. At this time, the best defense is not to have the smart card subjected to these conditions. Of course, that is not always possible, so there are some sleeves that purport to block radio waves and there are waterproof insulated wallets that can be used as interim measures until improvements in technology offset these threats. Threats that involve radiation have implications for data security. These are covered below.
- *Tampering.* Smart card processors can be physically protected by modifying the layout of the chip and hiding data pathways. Another option is to hide the real IC processes inside and throughout random false functions. Cards that show physical evidence of tampering can be delisted and retained and/or rendered useless. The use of holographs, hidden characters and other printing technologies can be used to make counterfeiting unprofitable. It should be noted that contactless smart cards present a greater risk than contact cards, in that wireless signals can be much more easily "bled and read," and deliberate jamming and other forms of electrical interference will increasingly be used as a weapon to disable both cards and readers. This activity could disrupt and destroy a digital economy, so technological solutions are needed.

6.2 Data Security

Data security involves *hot listing*, the electronic "recapture" and/or elimination of on-board encryption keys that authorize access, as well as the physical recovery of smart cards. The goal is to prevent data piracy, which involves data encryption, data validation and data theft (electronic trespassing, for example). Each presents its own challenges [19].

- *Hot listing and recovering cards*. Cards are generally considered the property of the card issuer, not the cardholder. Cards are de-authorized, or *hot-listed*, when the cardholder is no longer authorized to possess/use the card. Electronic hot-lists work, but only if the back office data transfer is timely.

For example, the card issuer may be slow to identify the new status, slow to key the status change into the system, or slow to communicate that change to affiliated networks or to physically transfer the status to offline card readers. If this does not happen, there is a window of opportunity for abuse of hot-listed smart cards. In the "old days" of contact smart cards, the card had to be inserted into a slot. Card readers could be programmed to retain the card; however, this was an unpopular option because cardholders were victimized both when they inadvertently abandoned their cards and when the card readers "ate" damaged, authorized or re-authorized cards. Contactless cards must be "de-listed" by readers that have specific read-write functionalities. It should be pointed out that having a card in a cardholder's possession leads to assumptions that it is valid. In addition, de-listing and retaining a card with multiple functionalities not directly associated with the original card issuer's primary reason for issuing the card will need to be addressed. Today, most readers do not have the capability to retain de-authorized cards.

- *Cryptography and cryptanalysis.* The strongest encryption methods available in the past were "1-time pads" that were randomly generated ciphertexts. For the message to be decrypted, the sender and the recipient had to have access to the *same* "1 time pad." Today's digital equivalent is the public key. More secure systems use both a public key and hidden (secret) key. The security problems with public key systems rest on the need to trade the key, often over unsecured lines. For this reason, the asymmetric methods are generally used for sensitive transactions. The security problems associated with asymmetric encryption systems are caused by conflicts between government and law enforcement (the need to know what is happening in order to prevent economic crimes such as money laundering (used to fund terrorism and other violent crimes)) and companies (the need to protect sensitive and proprietary data from global competitors who engage in economic and electronic espionage) [29]. Humans are eventually able to break most codes designed by humans, so we can expect that sufficient computing power will eventually be used to "break" most technology-driven encryption schemes. It is mainly a matter of resource allocation. For this reason, the security issues with encryption are aligned with public perception and the degree of acceptable economic/societal risk involved.

  For example, Bellcore Researchers threw the smart card market into disarray in 1995 when they announced that they had found a (theoretical) way to violate the security of smart card encryption. They claimed that criminals could heat the smart card (for example, in a microwave oven) or radiate it, thus tricking the smart card into making computational mistakes. By comparing actual

with anticipated values, criminals could use these mistakes to identify the *useful* patterns on the smart card that provide clues to the encryption keys and hidden information. They called this method "Cryptanalysis in the Presence of Hardware Faults" [6]. While subsequent work has not found a way to actualize this theoretical model, such publicity certainly should trigger industry concerns.

- *Data destruction.* A number of technologies could be used to invade systems and damage, destroy or steal data. These range from relatively low-technology applications such as visual and/or keystroke surveillance of cardholders engaged in smart card transactions to far more sophisticated techniques from Van Eck Phreaking to degaussing. Another threat deals with the common "cookie." Cookies were introduced by Netscape in 1994 *to solve the state retention program by introducing new headers to be carried over HTTP* [16]. Cookies are stored on the client side of memory (the user's computer), which allows a history of use to be developed and maintained. Cookies have been used to cross over and acquire personal data (a form of electronic trespassing) without the cardholder's (or card issuer's) knowledge. Research is underway to enable servers to track user's information-seeking behaviors *once they leave the server's site*. There are law enforcement, as well as marketing, benefits involved in finding ways to facilitate such link and pattern analysis. However, while the card belongs to the issuer, the data has been considered the property of the cardholder and the right to construct privacy fences around inquiries is considered a fundamental tenet of a democratic society.
- *Cross validation*. Identification verification is at the heart of the smart card's potential worth. Credit scoring systems use multiple databases, and score for data quality, to verify identity and data association. For this reason, smart card systems that rely on a single ID verification method, even a biometric one, are potentially dangerous. It is much easier to erase or change data in a single database than to do it in several dozen databases, especially where ownership of the databases involves multiple encryption schemes, various data sets and a number of organizations with competing agendas. While it is potentially expensive to include multiple biometric data sets, it is a false economy to assume that a single data set is (or will always be) sufficient. The best solution at this time is personal, portable biometrics where the cardholder's personal characteristics are compared to cross-validated data sets on the card and at remote sites at the point of use. This is extremely expensive and resource intensive.

7. Future Developments

At this time, there are three primary concerns that limit widespread market acceptance of the sophisticated features of smart cards: (1) available applications; (2) deployment costs; and (3) public concerns regarding such issues as data security (refer to the discussion on data security, above) and personal privacy.

- *Applications.* Applications require processing capability and memory capacity, which has been improved by the move from Assembly to C programming languages, use of Java, market preference for global standards and open architecture, and a growing interest in developing mini-databases.
- *Deployment costs.* While the relatively slow migration from magnetic stripe cards to smart cards continues to be based on financial factors (e.g., financial institutions with large investments in magnetic card systems are slow to invest in the new technology), the migration to more secure transaction processing systems represents regulatory differences. In countries where online transaction costs are low, online, real-time transactions are commonplace, and there is little economic incentive to migrate to more secured transactions. In countries where the cost of online transactions is high, off-line (batch) processing results in higher rates of card-related economic fraud, and there is more willingness to migrate to the more secure smart card technology.

  The "true" cost of any technology, however, includes both tangible costs and intangible benefits. Recent improvements in managerial and cost accounting practices enable decision makers to gain better insights into the return on investment of more secure access controls. Particularly after the attacks of September 11, 2001, smart card identification systems were investigated and there have been many new installations of these systems. Returns on investment do not always appear where they are anticipated. After implementing smart card technology, universities in the US noticed sharp reductions in armed robberies and vandalism of vending machines. Financial assistance to students was processed more quickly and involved fewer staff. American Express issued the Blue Card®, even though sophisticated applications were not yet in place. The expectation was that customers would "upgrade" their cards. The company found that simply having this technology available attracted many new customers [24].
- *Personal privacy.* Cardholder privacy becomes an issue where new developments allow applications to share PINs and user access histories become attached to individuals rather than specific terminals (mobile cookies). Economic conditions have not recently favored such investments. The slow rate of adoption is partly a result of psychological obstacles [34]. Research on National ID card program success/failure [83], for example, found that turf battles be-

tween government agencies in Australia over which agencies would have to migrate (update/replace their existing systems, a major investment) and which agency(ies) would not be required to make such an investment was sufficient to delay action until opponents of the card mounted successful opposition campaigns based on individual privacy concerns. The results have blocked issuance of an Australian National ID card for thirty years [41,93].

Even so, technical innovations continue to be important and new innovations are being brought to market. Most notably, these deal with (1) faster communication rates, (2) multiple operating systems on a single chip, and (3) the introduction of peripheral technologies for contactless cards [3,44,45,47–50,84].

- *Faster communication speeds.* The current bit rate for communication and anticollision is 106 kb/s. ISO Working Group 8 (WG8) has set a target ceiling of 847 kb/s for ISO 14443 and discussions concerning a related amendment has begun.
- *Multiple operating systems on a single chip*. Discussion has already begun regarding session-based switching of the chip operating system at the reader interface. This would enable the cardholder to add functionalities not offered by the card issuer; e.g., accounts with multiple banks and brokerage firms; secured access to facilities with proprietary operating systems; etc.
- *Peripheral card technologies*. WG8 is discussing a standard for the placement of a fingerprint sensor on the smart card, taking careful consideration of the placement's impact on the contactless antenna. WG I (Physical Characteristics and Test Methods for Identification Cards) is addressing additional systems on the card, including an interface specification for on-card displays, fingerprint sensors and keypads. Both WG4 (Integrated Circuit Cards with Contacts) and WG8 are monitoring these initiatives.

  There are three general categories of smart card applications that offer substantial benefits at the present time and/or in the near future: (1) *identification*, (2) *information storage*, and (3) *transaction processing*.
  - *Identification*. The locus of control that protects the economy, the nation and the critical infrastructure necessary to society's well-being is the ability to "*associate a particular individual with an identity*" [43] and validate that individual's right to engage in the requested transaction at the time, in the location and in the manner requested. Two specific approaches to identity verification are knowledge-based (what the individual knows, such as a personal identification number) and what the individual carries, such as a token (ID card, bank card or fob). The smart card provides an additional layer of security, in that personal biometrics (fingerprints, iris scans) can be compared with the individual who holds the card and knows the PIN.

Smart card-based identification has the potential to avoid both the fraud associated with the issue of driving licenses today and to meet the need for ID verification without offending an individual's personal beliefs. For example, verification of an individual processing a typical US driver's license is based on a photograph of the cardholder's face, but some individuals keep their faces covered in public. Therefore, photographs show only cloth head coverings. In addition, the licenses are valid for such extended periods of time that physical characteristics can change. High-tech, smart card-based driver's licenses with personal biometric data could use additional or alternate criteria to verify individual identities. In addition, such licenses could include relevant data, such as driving records or unpaid traffic fines [56,90].

– *Information storage.* There is a need to store ever-growing volumes of data, particularly personal demographic information [21]. The US healthcare industry is a prime example, where the Federal government has mandated electronic social benefits transfer and HIPAA compliance. In addition, there is a push to streamline business operations, including the automation of primarily clerical functions (such as fast admission for emergency room patients and linking unknown patients to their medical records in order to provide appropriate types and levels of care) [31]. For this reason, medical and healthcare information management systems are being upgraded. At the same time, there is a need to identify and provide services that attract new sources of revenue that are not tied to reimbursement schedules [83]. Among the potential applications of smart card technology in healthcare are: (1) automated hospital admissions; (2) transfer of medical records, drug prescriptions and insurance authorizations; and (3) technology information, such as individual kidney dialysis equipment settings [31]. Another new source of revenue is vending authorization (bedside delivery of upscale meals for those on a regular diet, for example). Again, there are significant social, political, legal/regulatory and economic issues that must be considered, which are beyond the scope of this discussion [76,84].
– *Transaction processing*. Smart card readers support both traditional and newer "in-home" ATM transactions and home shopping [80]. As we have seen with the Euro, currencies are merely points on a scale. The magnetic card has already proven to be a durable technology for international ATM transactions. Smart card technology could accelerate the decline of currency exchange and travelers' checks because the cards can carry currency in any foreign denomination and enable the cardholder to segregate both discretionary spending and job-related travel [21,38].

The newer contactless smart cards show great promise for improving the speed of access, e.g., transit passes and access to restricted facilities, such

as military bases and clean rooms. The "smart" nature of the card supports additional data capture for audit trail and security purposes. Today, smart cards are integrated into a wide range of telecommunications technologies. For example, GSM telephones can serve as portable ATM machines as well as locator beacons [33]. GSM telephones today use smart SIM (Subscriber Identity Modules) cards that can be encrypted [46,31]. There is a growing interest in eGovernment services. One possibility is the use of smart cards for eVoting. There is certainly a possibility that voter fraud could be reduced by linking the voter's biometrics with the casting of a ballot [43].

While it was once predicted that multifunction smart cards would dominate the smart card market in the near future [80], shipment data indicates the multifunction card has arrived. Standards appear to be relatively stable at the present time [3], but there are additional external considerations, such as competition for application space on the card and conflicts over transaction processing revenues. Several of the original companies involved in smart card innovation have since left all or part of the smart card industry. As a whole, however, the industry is healthy with many new niche markets and competitors. Most of the companies currently engaged in the development and sale of smart card applications are quickly developing expertise and/or the resources required [25,27,36,37,70,71,83].

Clearly, there continue to be "*software design, economics, liability and privacy concerns, consumer acceptance and . . . other political and personal issues*" [31]. There continue to be *vast implications that result from the contribution of smart cards in the "integration of commercial transactions, data warehousing and data mining*" [81]. Even so, smart card technologies are being used to improve the security of many transactions. As a result, smart card applications play an increasingly significant role in the nature and direction of information exchange [34,95–97].

Glossary of Terms

American National Standards Institute (ANSI) promotes national commerce in the form of interoperability facilitated through.

EFT—Electronic Funds Transfer. And debit/credit transaction that is processed by electronic means. Through **Reg E**, the Federal Reserve implemented the 1978 Electronic Funds Transfer Act of 1978.

EMV or VME Standard. Europay/MasterCard/Visa standard for contact smart cards, beginning with placement of chip on card (upper left) and specifics of the ABA stripe.

International Standards Organisation (ISO). Promotes interoperability through the development of recognized standards in order to improve international commercial exchange.

Contact. Smartchip cards that require the smartchip itself to be placed in physical contact with the reader device in a specific fashion. This contact is required for authentication/verification/authorization process.

Contactless/proximity. Smartchip cards that rely on a copper loop or other material embedded in the card. Readers sync with the low level radio frequency emissions of the card to facilitate fast authorizations for transactions that do not require the higher levels of authentication, i.e., inventory control, vehicle transit passes.

EPROM Electronically Programmable Read-Only Memory. Memory that is "write-once," commonly used in non-rechargeable stored value smart cards, i.e., prepaid simple memory phone cards.

EEPROM Electronically Erasable Programmable Read-Only Memory. Memory which can be written and erased multiple times. It is commonly used in reloadable stored value smart cards.

Farrington 7B. The type face specified in ISO standard 7811 for embossed characters on ID cards.

Firmware. The software that is written into the read-only memory that specifies the operation of a hardware component and is not alterable without changes to hardware configuration.

Hot card. An issued card that is no longer considered legitimate in the system, i.e., it has been reported lost or stolen, but has not been returned to the issuer.

Lasercard. An ID card technology that utilizes optical recording techniques to store data. Different from the magnetic recording techniques used on magnetic stripe cards, these cards are also referred to as optical memory cards.

RFm. An ID card in which the data from the card is transmitted to the reader via radio frequency emissions. RF cards are a common variety of proximity cards in that the card need not make physical contact with the reader.

RS232. A standardized interface that supports digital transmission of data between devices of various types.

Source: [13,14,85,86]

REFERENCES

[1] "A brief history of smart card technology: who, what, where, when and why", *Campus ID Report* **2** (1) (January 1997) 1, 3–4.
[2] "A reference guide to card-related ISO standards and committees", *Campus ID Report* **1** (5) (July 1996).
[3] Avisian, Inc., *Contactless Smart Card Technology for Physical Access Control*, 1 April 2002. An Avisian, Inc. Report, Unpublished White Paper.
[4] "Barcode basics: The ins and outs of these little black and white symbols", *Campus ID Report* **2** (3) (May 1996) 6–7.
[5] "Bar code basics", *Campus ID Report* **2** (3) (March/April 1997) 6–7.
[6] "Bellcore's shaky smart card 'threat model' shakes up card industry", *Campus ID Report* **1** (3) (May 1996) 6.
[7] Benini L., et al., "Energy-efficient data scrambling on memory processor interfaces", in: *ISLPED'03, Seoul, South Korea*, 25–27 August 2003, pp. 26–29.
[8] Berinato S., "Smart cards: The intelligent way to security", *Network Computing* **9** (9) (15 May 1998) 168.
[9] Block V., "Looking beyond chips, Motorola plans full range of smart card products", *American Banker* **62** (57) (25 March 1997) 14.
[10] Bolchini C., et al., "Logical and physical design issues for smart card databases", *ACM Trans. Inform. Systems* **21** (3) (July 2003) 254–285.
[11] Briney A., "A smart card for everyone?", *Information Security* (March 2002).
[12] "Campus ID cards in the library: The land of the sacred bar code", *Campus ID Report* **1** (6) (May 1996) 8.
[13] "Card industry lexicon", *Campus ID Report* **2** (3) (May 1996) 8.
[14] "Card industry lexicon: Understanding your campus card industry lingo", *Campus ID Report* **1** (4) (June 1996) 7.
[15] "Card Europe UK—background paper. Card Europe: The association for smart card and related industries" (online), http://www.cardeurope.demon.co.uk/rep1.htm, 1994. Accessed 29 April 2000.
[16] Chan A.T., Mobile cookies management on a smart card. Unpublished paper in review, 2003.
[17] Clendening J., "EDS announces US$1 billion in global IT business", *PR Newswire* (3 May 2001).
[18] Costlow T., "Major players bet on new smart card standard", *Electronic Engineering Times* **965** (4 August 1997) 6.
[19] Craig B., "Resisting electronic payment systems: burning down the house?" *Economic Commentary* (July 1999).
[20] Croghan L., "Chase, Citi put their money on smart card pilot program", *Crain's New York Business* **13** (39) (29 September 1997) 1.
[21] Cross R., "Smart cards for the intelligent shopper", *Direct Marketing* **58** (12) (April 1996) 30–34.
[22] "D2T2: Printing on plastic", *Campus ID Report* **1** (4) (June 1996) 8–9.

[23] Davis D., "Chip cards battle bar codes for big ID project", *Card Technology* (March 2002).
[24] Donoghue J.A., "White hats/black hats: A pre-screened group of passengers may make it easier to concentrate security efforts on the rest, but efforts to construct a 'trusted traveler' system are off to a slow, uncoordinated start", *Air Transport World* **38** (3) (March 2002).
[25] Eedes J., "Growth companies. Leaders of the technology pack", *Financial Mail* (26 October 2001).
[26] "Eelskin wallets and other misconceptions about magnetic stripes", *Campus ID Report* **1** (8) (September 1996) 1, 5.
[27] "Electronic services use set to explode", *Bank Systems & Technology* **34** (8) (August 1997) 15.
[28] Elliot J., "The one-card trick—multi-application smart card e-commerce prototypes", *Comput. Control Engrg. J.* **10** (3) (June 1999) 121–128.
[29] "Encryption part II: Certificates and certificate authorities", *Campus ID Report* **1** (6) (August 1996) 1, 6.
[30] "Evaluating the VISA cash pilot in Atlanta: A campus view", *Campus ID Report* **1** (6) (August 1996) 1, 5.
[31] Fancher C.H., "Smart cards", *Scientific American* (1 August 1996) (online), http://www.sciam.com/0896issue/0896fancher.html. Accessed 1 April 2000.
[32] "FDIC examines stores value: Should it be treated as a deposit?" *Campus ID Report* **1** (7) (September 1996) 1, 5.
[33] Fletcher P., "Europe holds a winning hand with smart cards", *Electronic Design* **47** (1) (11 January 1999) 106.
[34] Flohr U., "The smart card invasion", *Byte* **23** (1) (January 1998) 76.
[35] Ganeson P., et al., "Analyzing and modeling encryption overhead for sensor network nodes", in: *WSNA'03, San Diego, California, USA*, 2003, pp. 151–159.
[36] Gjertsen L.A., "Insurers turn to stored value cards", *American Banker* **6** (116) (2001).
[37] Goedert J., *Health data management* (February 2000).
[38] Gray D.F., "Euro spurs I-commerce uptake", *InfoWorld* **21** (15) (12 April 1999) 70.
[39] Grossschadl J., "Architectural support for long integer modulo arithmetic on RISC-based smart cards", *Internat. J. High Performance Comput. Appl.* **17** (2) (Summer 2003) 135–146.
[40] Guyon J., "Smart plastic", *Fortune* **136** (7) (13 October 1997) 56.
[41] Hempel C., "National ID card stirs a world of debate", *Knight-Ridder Tribune Business News* (19 December 2001).
[42] Husemann D., "The smart card: don't leave home without it", *IEEE Concurrency* **7** (2) (April–June 1999) 24–27.
[43] Jain A., Hong L., Pankanti S., "Biometric identification", *Communications of The ACM* **43** (2) (February 2000) 90–98.
[44] Kutler J., "Visa-promoted CEPS making inroads in Europe, Asia", *American Banker* **163** (226) (25 November 1998) 11.
[45] Kutler J., "Cell phone-smart card hookup eyed for US after winning over Europe", *American Banker* **163** (212) (4 November 1998) 1.

[46] Kutler J., "Card giants using EMV as a stepping stone", *American Banker* **164** (114) (16 June 1999) 11.
[47] Kutler J., "Java gets pats on back from card businesses in Belgium and France", *American Banker* **164** (61) (31 March 1999) 16.
[48] Kutler J., "Visa regional operation adopts six-year plan for smart card conversion", *American Banker* **164** (109) (9 June 1999) 13.
[49] Kutler J., "A boost from Europe for card readers", *American Banker* **164** (197) (13 October 1999) 18.
[50] Ladendorf K., "Americans wishing up to 'smart card' technology", *Knight-Ridder Tribune Business News* (1 October 2001).
[51] Leung A., "Smart cards seem a sure bet InfoWorld.com" (online), http://unix.idg.net/crd_smart_69240.html, 8 March 1999. Accessed 29 April 2000.
[52] Lu C., dos Santos A.L.M., Pimental F.R., "Implementation of fast RSA key generation on smart cards", in: *SAC 2002, Madrid, Spain*, 2002, pp. 214–220.
[53] "Magnetic stripes: Track by track", *Campus ID Report* **1** (1) (March 1996) 6–7.
[54] "Mag stripe stored value", *Campus ID Report* **1** (10) (November 1996).
[55] Marcial G.G., "From cubic: The ID cards of tomorrow", *Business Week* (19 November 2001).
[56] McGregor O., McCance O., "Use of chips smart?; Debate rages oven licenses", Richmond Times-Dispatch (19 February 2002) in *Knight-Rider Tribune Business News*.
[57] Moore A., "Highly robust biometric smart card design", in: *IEEE Transactions on Consumer Electronics*, vol. 46, 2000.
[58] Morrison D.J., Quella J.A., "Pattern thinking: Cutting through the chaos", *Marketing Management* **8** (4) (Winter 1999) 16–22.
[59] Muller H., "Europe's Hi-Tech edge", *Time* (31 January 2000) 28–31.
[60] Nairn G., "Survey—FT-IT: Fragmentation hinders medical market growth", *Financial Times Surveys Edition* (18 April 2001).
[61] Anonymous, "New homeland security department aims for IT compatibility", *Newsbytes* (7 June 2002).
[62] Newman D., "PKI: Build, buy or bust?", *NetworkWorld* (10 December 2001).
[63] "Olympic visa cash trial to be the highest profile test of a stored value card to date", *Campus ID Report* **1** (4) (June 1996) 11.
[64] Orr T.L., "FEMA responds with technology", *Government Computer News* **20** (8) (16 April 2001).
[65] OSCIE, Open smart card infrastructure for Europe v 2; Volume 6: Contactless technology; Part 1: White paper on requirements for the interoperability of contactless cards. Issued by eESC TB 8 Contactless Smart Cards, March 2003.
[66] "PAR technology subsidiary awarded $5.1 million government contract", *Business Wire* (31 May 2001).
[67] Phillips A., "Poised to take off", *Electronic Engineering Times Hot Markets Special Report* (October 2000) 22.
[68] Priisalu J., "Frequently asked questions list, Estonian institute of cybernetics" (online), http://www.ioc.ee/atsc/faq.html, 4 July 1995. Accessed 30 April 2000.

[69] Proffitt D., "Travelers wise up, use smart card", *Business Journal (Phoenix)* **16** (23) (5 April 1996) 29–30.
[70] Radice C., "Smart cards hype or solutions?", *Grocery Headquarters* **68** (1) (January 2002).
[71] Redd L., "Improving your return on IT", *Health Forum* (2 July 2002).
[72] "Regulation E and Campus Cards", *Campus ID Report* **1** (2) (April 1996) 1, 5.
[73] Reid K., "Pass and pay technology", *National Petroleum News* **92** (2) (February 2000) 32–38.
[74] "RFID cards: How proximity cards operate", *Campus ID Report* **1** (5) (July 1996) 8.
[75] Rogers E.M., *Diffusion of Innovations*, third ed., Free Press, New York, 1983.
[76] Rogers A., "European Parliament gains ground in health", *Lancet* **347** (9009) (27 April 1996) 1180.
[77] Sanchez-Reillo R., "Securing information and operations in a smart card through biometrics", *IEEE* (2000).
[78] Sanchez-Reillo R., "Smart card information and operations using biometrics", *IEEE Aerospace and Electronic Systems Magazine* (April 2001) 52–55.
[79] Sanchez J.S., "Disaster protection for financial data", *Financial Executive* **17** (9) (December 2001).
[80] Schacklett M., "These business trends will shape the future of e-commerce", *Union Magazine* (January 2000) 14–15.
[81] Shelfer K.M., "Intersection of knowledge management and competitive intelligence: Smart cards and electronic commerce", in: *Knowledge Management for the Information Professional*, Information Today, Medford, NJ, 1999.
[82] Shelfer K.M., Procaccino J.D., *Communications of the ACM* **45** (7) (July 2002) 83–88.
[83] Shelfer K.M., Procaccino J.D., National ID card programs: Requirements engineering, Preliminary results, unpublished paper.
[84] Shelfer K.M., Procaccino J., Smart health cards: Generating new revenues from old services, unpublished paper.
[85] "Smart cards 101: Different chips, different terms", *Campus ID Report* **1** (2) (April 1996) 10–11.
[86] "Smart cards 102: Basic operations of an IC card", *Campus ID Report* **1** (9) (November 1996) 5.
[87] Spurgeon B., "Big brother aside, smart ID cards are making global converts", *International Herald Tribune* (16 November 2001).
[88] Thaddeus J., "Disaster strategy: Bring continuity from calamity", *Computerworld* **35** (7) (12 February 2001).
[89] "The ins and outs of encryption", *Campus ID Report* **1** (5) (May 1996) 1, 4–5.
[90] Thibodeau P., "License bill could create IT headaches", *Computerworld* **36** (17) (22 April 2002).
[91] Tillett S., "INS to issue digital green card", *Federal Computer Week* (8 September 1997).
[92] "Time to get smart. (Smart cards and EMV)", *Cards International* **14** (7 May 2003).
[93] "Understanding the Buckley amendment: Is your campus card violating privacy laws?", *Campus ID Report* **1** (2) (April 1996).

[94] "Understanding the 10-digit ISO number", *Campus ID Report* **2** (2) (February 1997) 1, 5.
[95] Webb C.L., "Tech firms still waiting for floodgates to open", *Newsbytes* (15 April 2002).
[96] Weinberg N., "Scare tactics", *Forbes* (4 March 2002).
[97] Whitford M., "Unlocking the potential", *Hotel & Motel Management* **214** (3) (15 February 1999) 61.

Shotgun Sequence Assembly

MIHAI POP

The Institute for Genomic Research (TIGR)
Rockville, MD 20850
USA
mpop@tigr.org

Abstract

Shotgun sequencing is the most widely used technique for determining the DNA sequence of organisms. It involves breaking up the DNA into many small pieces that can be read by automated sequencing machines, then piecing together the original genome using specialized software programs called assemblers. Due to the large amounts of data being generated and to the complex structure of most organisms' genomes, successful assembly programs rely on sophisticated algorithms based on knowledge from such diverse fields as statistics, graph theory, computer science, and computer engineering. Throughout this chapter we will describe the main computational challenges imposed by the shotgun sequencing method, and survey the most widely used assembly algorithms.

ADVANCES IN COMPUTERS, VOL. 60
ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60006-9

1. Introduction

In 1982 Fred Sanger developed a new technique called **shotgun sequencing** and proved its worth by sequencing the complete genome of the bacteriophage Lambda [1]. This technique attempted to overcome the limitations of sequencing technologies by breaking up the DNA at random. Sequencing techniques were only able to "read" several hundred nucleotides at a time. The resulting pieces were **assembled** together based on the similarity between pieces derived from the same section of the original DNA molecule. The large amount of data produced by shotgun sequencing made it necessary to utilize computer programs to assist the assembly [2,3]. Despite continued improvements in sequencing technology and the development of specialized assembly programs, it was unclear whether shotgun sequencing could be used to sequence genomes larger than those of viruses (typically 5000–100,000 nucleotides). For larger genomes it was thought that the complexity of the task would pose an insurmountable challenge to any computer program.

In 1995, however, researchers at The Institute for Genomic Research (TIGR) successfully used the shotgun sequencing technique to decipher the complete genome of the bacterium *Haemophilus influenzae* [4]. The sequencing of this 1.83 million base pair genome required the development of a specialized assembly program [5] as well as painstaking laboratory efforts to complete those regions that could not correctly be assembled by the software. The success of the *Haemophilus* project started a genomics revolution with the number of genomes being sequenced every year increasing at an exponential rate. At the moment the genomes of more than 1000 viruses, 100 bacteria, and several eukaryotes have been completed, while multiple other projects are well on the way to completion. In parallel with the large amounts of genomic data becoming available, the genomic revolution led to the birth of a new field—bioinformatics—bringing together an eclectic mix of scientific fields such as computer science and engineering, mathematics, physics, chemistry, and biology.

Critics of the shotgun sequencing approach continued to question its applicability to large genomes despite obvious successes in sequencing bacterial genomes. They argued the technique would be impractical in the case of large eukaryotic genomes because **repeats**—stretches of DNA that occur in two or more copies within the

genome—would hopelessly confuse any assembler [6]. The standard procedure for handling large genomes was a hierarchical approach involving breaking up the DNA into large (50–150 kbp) pieces cloned in bacterial artificial chromosomes (BACs), and then sequencing each BAC through the shotgun method. Most such criticism was silenced in 2000 by the successful assembly at Celera of the genome of *Drosophila melanogaster* [7] from whole-genome shotgun (**WGS**) data. The assembly was performed with a new assembler [8] designed to handle the specific problems involved in assembling large complex genomes. The researchers from Celera went on to assemble the human genome using the same whole-genome shotgun sequencing technique [9]. Their results were published simultaneously with those from the International Human Genome Sequencing Consortium, who used the traditional hierarchical method [10]. Independent studies [11,12] later showed that the two assemblies produced similar results and many of the differences between them could be explained by the draft-level quality of the data. The applicability of the WGS method to large genomes was thus proven though some continue to argue the validity of Celera's results (some opinions on this topic are presented in [13–17]).

Celera's success combined with the cost advantages of the WGS technique—Celera sequenced and assembled the human genome in a little over a year while the international consortium's efforts had been going on for more than 10 years—renewed interest in the WGS method and led to the development of several WGS assembly programs: Arachne [18,19] at the Whitehead Institute, Phusion [20] at the Sanger Institute, Atlas [21] at the Baylor Human Genome Sequencing Center, and Jazz [22] at the DOE Joint Genome Institute. Most current sequencing projects have opted for a WGS approach instead of the hierarchical approach. For example the sequencing of the mouse [23], rat [24], dog [25], puffer fish [22], and sea squirt [26] all follow the WGS strategy.

The current issue of debate is the suitability of whole-genome shotgun sequencing as the starting point in the efforts to obtain the complete sequence for a genome. All sequencing strategies start by building a backbone, or rough draft, of the genome whose gaps need to be filled in through further laboratory experiments. It is still not clear which sequencing strategy will ultimately be the most efficient in obtaining the complete sequence of an organism, especially as none of the large eukaryotic projects have yet been finished, except for the 100 Mbp genome of the nematode *Caenorhabditis elegans*, finished in October 2002. (The genomes of *Drosophila melanogaster* and human are expected to be mostly finished before the end of 2003.)

Despite significant differences in the overall structuring of the sequencing process, all sequencing strategies rely on shotgun sequencing as a basic component. The reader is referred to [27,28] for an in-depth discussion of current approaches to sequencing. The following sections represent a description of the shotgun sequencing technique, with a emphasis on the algorithmic challenges imposed by this technique.

2. Shotgun Sequencing Overview

The process of shotgun sequencing starts by physically breaking up the DNA molecule into millions of random **fragments**. The fragments are then inserted into **cloning vectors**[1] in order to **amplify** the DNA to levels needed by the sequencing reactions. Commonly used cloning vectors are **plasmids** (circular pieces of DNA) that are then grown in the *Escherichia coli* bacterium. The plasmids' DNA sequence is engineered to enable sequencing reactions to proceed into the inserted fragments. The ends of each of the original fragments can thus be sequenced as shown in Fig. 1.

The random fragments (also called **inserts**) are usually organized into several **libraries** consisting of fragments of similar size. A typical sequencing project uses at least two such libraries: a small one (fragments ranging from 2 to 4 kbp) and a large one (8 to 10 kbp). In addition, large genome projects may also include several larger libraries such as **fosmid**[2] (40–42 kbp) or **BAC**[3] (50–150 kbp) libraries. Current sequencing technologies can only "read" between 600 and 1000 base pairs of

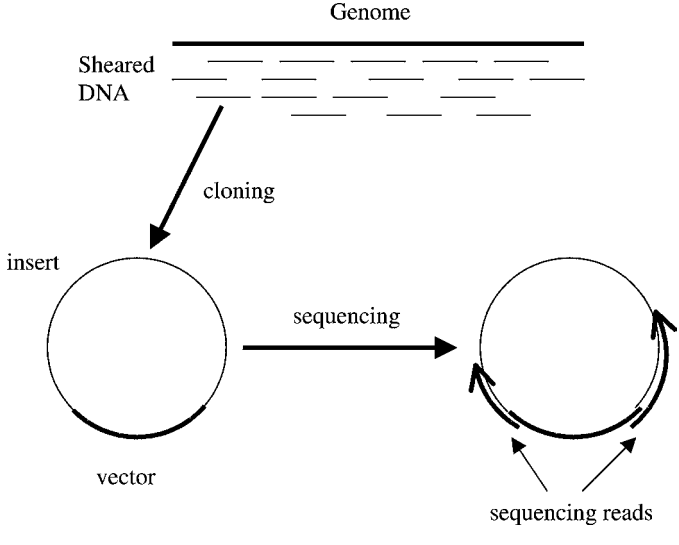


FIG. 1. Shotgun sequencing process.

[1]**cloning vectors**—DNA molecule from a specific host (virus, bacterium, or another higher organism) into which a DNA fragment can be inserted such that it will be replicated by the host organism.

[2]**fosmid**—cloning vector that accepts DNA inserts of about 40 kbp with a very tight size distribution.

[3]**bacterial artificial chromosome (BAC)**—cloning vector used to clone large DNA fragments (100–300 kbp).

DNA, and, therefore, the middle of the fragments remains unsequenced. This leads to pairs of reads (also called **mate-pairs**), obtained from opposite ends of a same fragment, which are naturally related. Such pairing information is essential to all modern assembly algorithms.

In the most basic formulation, the task of an assembler is to connect together the reads produced by the shotgun method in order to recover the original DNA sequence in a process not unlike putting together a jigsaw puzzle. The assembler uses the sequence similarity between two reads to infer that they may originate from the same region of the genome. This assumption is incorrect in the case of **repeats**—stretches of DNA that occur in multiple identical or near-identical copies throughout the genome—where the assembler may incorrectly combine reads coming from distinct copies of a particular repeat. The reader familiar with jigsaw puzzles has undoubtedly encountered this situation when trying to put together pieces of sky. In the best case repeats lead the assembler to generate more than one contiguous section of DNA (called a **contig**), while in the worst case the assembler may incorrectly reconstruct the original DNA. For example in Fig. 2 the repeat R tricks the assembler into swapping the order of sections I and II of the genome. The read pairing information can help the assembler detect and correct such errors, for example in the previous example the rearrangement invalidates the link between reads A and B.

Even in the absence of repeats, however, the output of the assembler may consist of more than one contig. This phenomenon can be explained with a simple analogy. Imagine a sidewalk as it starts to rain. As droplets fall, the sidewalk becomes increasingly wet, yet many spots remain dry for a while. Similarly, as the fragments are being sequenced, the randomness of the shearing process leads to sections of the original DNA not represented in the collection of reads. Therefore the best possible

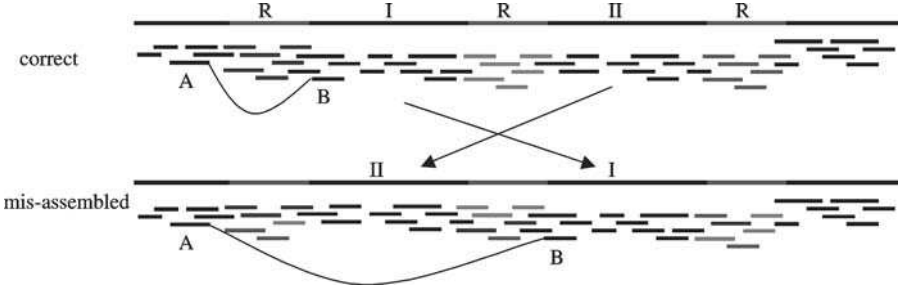


FIG. 2. Genome rearrangement around a repeat. The top section of the picture represents the correct layout. Each copy of repeat R is colored in a different color. The bottom image shows a possible misassembly due to repeat R.

assembly will consist in a collection of contigs with gaps in between representing those DNA stretches not present in the reads.

This phenomenon was initially analyzed by Lander and Waterman [29]. They approximate the "arrival" of N reads of equal length L along a genome of length G as a Poisson process. Considering the reads in order of their arrival along the genome (from left to right), the number of contigs is the same as the number of reads that do not **overlap** the read immediately following them. The notion of overlap warrants a further explanation: two reads are said to overlap if their relationship can be detected by an assembler. In the case of most assembly programs the overlap cannot be detected unless the reads share more than 40 base pairs. Lander and Waterman model this phenomenon through the parameter θ—the fraction of L by which two reads must overlap in order for the overlap to be detected by the program. The number of contigs can thus be computed by analyzing the probability that a particular read r occurs at the right end of a contig, i.e., the reads starting to the right of r do not overlap it. The probability of r starting at a particular base along the genome, assuming a perfectly uniform sampling process, is $\alpha = \frac{N}{G}$. Thus the probability of finding a read that does not overlap any of the reads starting after it (a right-end of a contig) is $\alpha(1-\alpha)^{L\sigma}$ where σ is $(1-\theta)$—the fraction of a read not involved in the minimum detectable overlap. The number of contigs represents the number of times we can exit such a read without detecting any overlaps and can be approximated by $Ne^{-\frac{NL}{G}\sigma}$. The fraction $\frac{NL}{G}$ represents the amount of over-sampling of the genome with reads, a number usually called **coverage**. Figure 3 shows the typical dependency between the number and length of contigs and coverage. The numbers are based on a genome of 5 Mbp covered by reads of length 600 bp. The graph matches intuition: at first the number of contigs increases as most contigs contain one single read. As the number of sequenced reads increases, so do the chances that reads will overlap to form larger contigs. After a certain point ($1.08\times$) the increase of coverage leads to a decrease in the number of contigs, and an increase of the expected contig length.

It is important to note at this point that the shotgun sequencing process is inherently inefficient. Lander–Waterman statistics indicate that in order to almost completely cover a genome one needs to sequence enough reads to cover the genome more than 8 times. At this level of coverage 99.9% of all bases are contained in at least one read. We are thus forced to sample each base 8 times when two separate sequencing experiments would be sufficient (the two-fold redundancy is necessary to reduce the effect of sequencing errors). The remaining bases—the **gaps** between contigs—will need to be determined through targeted sequencing experiments. We are thus faced with the basic trade-off of any shotgun sequencing project: the proper ratio between the random and targeted sequencing reactions that minimizes the cost of sequencing a genome to completion. The random sequencing phase of a project is highly automated and therefore very efficient in terms of per-read cost (currently less

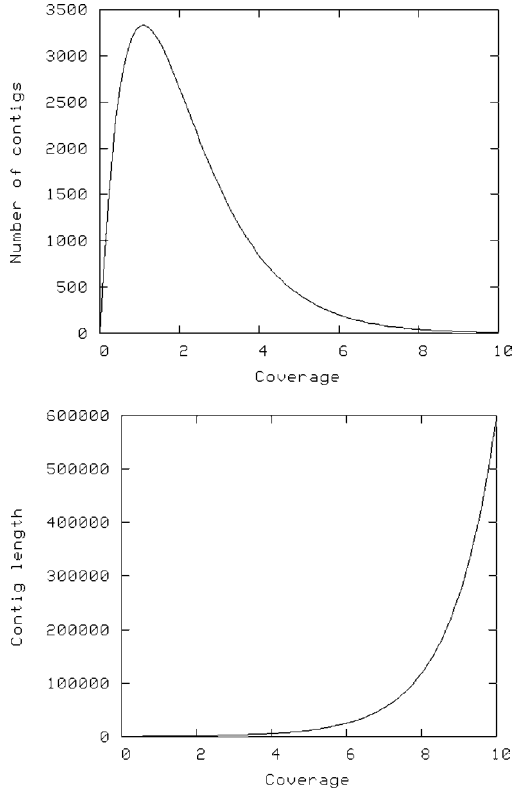


FIG. 3. Dependence of number and length of contigs on genome coverage.

than $1.00 per read, and still falling). Targeted sequencing, however, involves direct human intervention and is therefore much costlier. The genome size is another factor in this decision: a mammalian genome is 1000 times larger than a bacterial genome, and correspondingly more expensive. Several attempts [30,31] have been made to determine the most cost-effective strategy for sequencing a genome to completion, yet a universally accepted solution is not yet available.

In the previous paragraphs we discussed the situation when the particular genome being sequenced is considered given and we are attempting to model an idealized shotgun sequencing experiment. Li and Waterman [32] address the dual question: given a set of shotgun reads, what conclusions can we draw about the structure of the genome being sequenced? Specifically, they attempt to determine the length and repeat content of the original genome that best explains the observed set of shotgun reads. This problem has great importance in practical applications since the specific

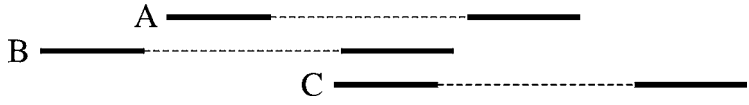


FIG. 4. Fragment overlaps can only be detected when corresponding reads (represented by the thick segments) overlap. The relationship between fragments A and B or fragments A and C cannot be detected. The overlap between fragments B and C is implied by the overlap of their end-reads.

characteristic of the genome being sequenced are often unknown before sequencing. Moreover, a solution to this problem can provide an invaluable method for quality control during the shotgun process, by highlighting discrepancies between the expected structure of the genome and the information inferred from the shotgun data. Li and Waterman's approach is based on analyzing the frequencies of occurrence of all l-tuples (strings of length l) present in the reads then matching the observed distribution to the expected values using a standard Expectation–Maximization (EM) algorithm. They also provide an algorithm for characterizing the structure of the repeats present in the genome.

The Lander–Waterman statistics described above represent an approximation of a true sequencing project. They assume all reads have the same length L, a case not encountered in practice. Furthermore, as described above, sequencing reads represent a fraction of the shotgun fragments, as only the ends of the fragments are being sequenced. The overlap between two fragments cannot be determined unless the sequenced ends overlap. An example is shown in Fig. 4. These limitations of the Lander–Waterman model were addressed by Arratia et al. [33]. They examine a model where the overlap between two fragments can only be detected at particular "anchors" distributed along the genome. The concept of anchors is very general. It may include short **restriction sites**[4] commonly used in physical mapping experiments (the original focus of these statistical analyses), however it also encompasses sequencing reads. Furthermore, the authors address the issue of variable fragment lengths by assuming a particular distribution of fragment sizes. They obtain an interesting result: if fragment lengths are sampled from a normal distribution, an increase in the standard deviation for a particular mean leads to three events: (i) the expected number of contigs decreases; (ii) the expected contig length increases, and (iii) the expected fraction of the genome covered by contigs increases. Variability in read length therefore has a beneficial effect on the outcome of a shotgun-sequencing experiment.

Arratia et al.'s main contribution however is to provide a first statistical analysis of **scaffolds,** a concept essential to all modern assembly algorithms. The relation-

[4]**restriction sites**—short stretches of DNA that are recognized by specialized proteins (**restriction enzymes**) which cut the DNA at these sites.

FIG. 5. Relationship between contigs as inferred from read pairing information.

ship between the two sequence reads derived from the same fragment can provide useful linking information between the two contigs containing the reads. The estimated length of the fragment provides an estimate for the size of the gap between the contigs, while the orientation of the reads within the contigs determines the relative orientation of the two contigs as shown in Fig. 5. Note that the orientation attribute of a read or contig is a direct consequence of the double-strand structure of DNA. Each strand has an implicit orientation determined by the direction in which DNA is replicated, always from one end (denoted **5'**) to the other (denoted **3'**). The two complementary strands have opposite orientations, however the global orientations defined by the DNA molecule that is being sequenced are lost through the shearing process. For any particular read it is therefore impossible to determine from which strand of the original molecule it originated. Assembly programs reconstruct one of the two strands (the other one is simply the reverse complement of the first) though not necessarily the same strand for all the contigs. It is therefore important to determine the relative "orientation" of the contigs, specifically whether two contigs represent the same DNA strand, or they come from opposite strands. The contigs thus placed in a consistent order and orientation form a **scaffold**—term first introduced by Roach et al. in [34]. An extension of the Lander–Waterman statistics in the case of end-sequenced fragments was presented by Port et al. in [35]. They considered a collection of fixed-length fragments and analyzed a simplified definition of scaffolds, specifically scaffolds that can be greedily generated by iteratively attaching fragments to their rightmost end. They ignore the effect of transitively inferred relationships, i.e., situations when the overlap between two fragments can only be inferred from their overlaps with a third fragment (see Fig. 6). Yeh et al. (Yeh et al. manuscript submitted [36]) extended this analysis to the general case.

Our inability to sequence more than just the ends of each fragment leads to an interesting situation. Scientists must sequence enough fragments to guarantee that most of the genome will be represented by sequencing reads. As described by the Lander–Waterman statistics, one needs to sequence roughly 8 times the size of the genome in order to guarantee that almost all bases (99.9%) are contained in at least one of the sequencing reads. The reads, however, represent only a fraction of the fragments. The number of fragments needed to provide the required $8\times$ coverage by reads, implies a "fragment" or "clone" coverage of the genome greater than that by reads alone. Intuitively, each fragment contributes to the overall coverage both the

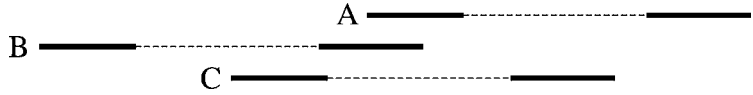


FIG. 6. The overlap between fragments A and C can be inferred from their overlaps with fragment B even though none of their reads overlap.

section covered by reads, and the unsequenced portion of the fragment. To clarify this phenomenon we provide a simple example. Assume a 1 Mbp genome covered by 2 kbp fragments. Also assume that the sequencing reads are all 500 bp long. Sequencing at 8× read coverage requires 16,000 reads, and therefore 8000 fragments. These fragments cover 16 Mbp, and thus cover the genome 16 times (16× fragment coverage). This effect is even more pronounced in the case of longer fragments. Mathematically, the relationship between fragment coverage (*fc*) and read coverage (*rc*) is

$$fc = rc \cdot \frac{flen}{2 \cdot rlen}$$

where *flen* and *rlen* are the average fragment and read lengths, respectively. Applying the Lander–Waterman statistics to the fragments shows that virtually every base of the genome is contained in at least one fragment. This is a very important fact since it implies that those bases not represented in the collection of reads will likely be covered by at least one fragment. The fragments can later be used as a substrate for specialized sequencing reactions targeting the unsequenced bases.

Port et al. also introduced a statistical framework for handling the case of non-uniform sampling of the genome, a commonly encountered situation. Often times certain characteristics of the DNA fragments, such as short stretches of repetitive DNA, prevent their efficient replication in the cloning vector [37]. As a result, some sections of the genome are represented poorly or not at all in the fragment collection. Port et al. [35] were the first to mathematically address these situations by modeling the cloning bias as an inhomogeneous Poisson process. It is important to discuss some of the implications of non-random libraries on assembly software. Let us assume that the genome being sequenced contains a section (marked T in Fig. 7) that is toxic to the cloning vector, and therefore not present in the collection of sequenced reads. T's toxicity implies that the library does not contain any fragment that encompasses a significant portion of the region, therefore T remains largely unsequenced, except for the ends being covered by reads from un-affected fragments. The more important effect, however, is the absence of linking information across this region, since the linking data is inferred from the fragments. The toxic region thus prevents the formation of scaffolds that span it, which in turn means that any assembly will break at this region.

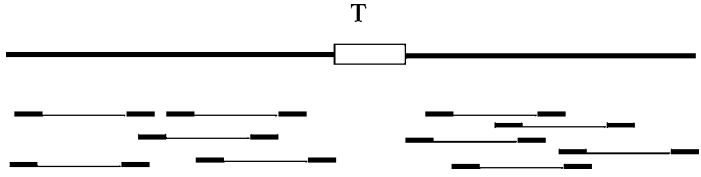


FIG. 7. Effect of toxic region on fragment coverage.

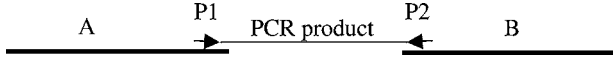


FIG. 8. Contig adjacency as inferred from PCR reaction. The length of the product between primers P1 and P2 provides an estimate on the size of the gap between contigs A and B.

The relative order and orientation of the resulting scaffolds is generally determined through laborious experimental techniques. The basic idea is to use of the **polymerase chain reaction**[5] (**PCR**) [38] technique to amplify DNA directly from the genome, without using a cloning vector. Scientists design PCR **primers**[6] (short stretches of DNA that provide a starting point for the PCR reactions) at the ends of all scaffolds, then perform a set of PCR reactions in order to find pairs of primers that are related. The technical issues in running PCR reactions are complex, however the basic idea is that the PCR reaction will amplify the region between two primers if they are actually adjacent in the genome. The size of the **PCR product**[7] (the amplified section) provides insight in the actual distance between the primers (see Fig. 8). In the case of an assembly with $\frac{N}{2}$ scaffolds we need to determine the adjacency information for N different scaffold ends. In essence we must perform $N(N-1)$ PCR-comparisons to determine which ends are next to each other. For many bacterial genomes N is less than 100, however a typical eukaryotic genome may yield hundreds or thousands of scaffolds. For simple genomes we need to perform hundreds of PCR reactions, while for large genomes the technique becomes practically infeasible and we therefore need additional information to order and orient the scaffolds. Nonetheless, this experimental technique is widely used in practice and efforts were made to improve it. A related technique called **multiplex PCR**[8] [39] was developed that allows scientists to pool together several primers (up to about 40) and

[5]**polymerase chain reaction (PCR)**—laboratory technique through which a DNA segment is rapidly replicated.

[6]**PCR primer**—short (tens of base pairs) stretch of DNA that provides a starting point for the PCR reaction.

[7]**PCR product**—the DNA segment amplified by the PCR reaction.

[8]**multiplex PCR**—technique that allows multiple PCR reactions to be performed at the same time.

effectively test all possible adjacencies of the primers in the pool in one step. The result of this procedure is a certificate that one or more pairs of primers are adjacent, unfortunately no information is provided as to which pairs were successfully tested. Further tests are necessary to determine exactly which primers were adjacent. Tettelin et al. [40] introduced the following problem: given N PCR primers, and a limit K on the number of primers that can be included in a single multiplex pool, determine the minimum number of reactions needed to check all possible primer adjacencies. The authors provide a solution that requires $P + \sqrt{P}$ reactions where P is a perfect square with the property that $\frac{N}{P} < K$. The authors notice, however, that while this solution optimizes the number of reactions, it does not minimize the number of laboratory operations needed to perform these reactions. For each pool scientists need to pipette all the primers into a reaction tube. Because this pipetting operation is the source of most laboratory errors, Tettelin et al. proposed a solution that attempts to minimize the number of pipetting steps. Their solution, an algorithm called **Pipette Optimized Multiplex-PCR (POMP)** requires $2N - \sqrt{N}$ pipetting operations, however it cannot be proven optimal. After a first step of multiplex-PCR reactions, the results of the POMP method need to be deconvoluted in order to obtain the actual pairs of adjacent primers, leading to the need for additional reactions. They do not present a theoretical analysis of the number of such additional reactions required by the technique. Beigel et al. [41] extended the theoretical framework by addressing the problem of correctly identifying all primer adjacencies in a reaction-efficient manner. They prove a lower bound of $\Omega(n \log n)$ reactions needed to determine all the adjacencies and describe a multi-round solution to the problem that is within a constant factor of the optimal bound. Alon et al. [42] further extend the analysis and propose a probabilistic non-adaptive solution that matches the lower bound to within a constant factor. Non-adaptive solutions are useful as they enable the automation of the procedure.

This scaffold ordering step is part of the final stage of a sequencing project called **gap closure**[9] or **finishing**. This labor-intensive stage encompasses a variety of bioinformatics and laboratory techniques meant to "fill in" the remaining gaps in the genome map, as well as to detect and correct mis-assembled repeats. The finishing stage is one of the most expensive stages of a sequencing project and thus can benefit from the support of specialized software. As an example, we refer the reader to several papers describing the finishing support tools included in the commonly used phred-phrap package [43–45]. Furthermore, Czabarka et al. [30] mathematically analyze the closure process and present optimal solutions (in terms of the number of finishing reactions needed) in the context of several theoretical models of the finishing stage. It is worth observing the synergy between the development of shotgun

[9] **gap closure** or **finishing**—process through which the gaps between the contigs produced by the assembler are closed through targeted sequencing reactions.

sequence assembly programs and the experience of human experts involved in the finishing process. Most modern assemblers include features useful to the finishing process by providing the types of information needed to guide additional laboratory experiments. As an example, the Euler package contains a module that designs the experiments needed to resolve over collapsed repeats [46]. TIGR Assembler [5] is, at the moment, unique in its ability to allow the users to "freeze" certain assemblies and thus manually guide the assembly process. At the same time, assembly programs use criteria developed by finishing experts to detect and correct mis-assemblies. The Arachne assembler [19] includes such a module.

The following sections will describe in detail the various problems associated with shotgun sequence assembly. Section 3 is devoted to the discussion of the most common assembly strategies at a conceptual level. Practical details of implementing specific tasks required by assembly program are discussed in Section 4. Finally, Section 5 will describe several new challenges to assembly programs.

3. Assembly Paradigms

In its most general form the sequence assembly problem involves reconstructing the genome from the shotgun reads based on sequence similarity alone. This problem can be further decomposed into two problems: the **mapping** or **layout** problem, in which all reads need to be positioned correctly in the genome, and the **consensus** problem, in which the contiguous DNA sequence of the genome is computed. It can be easily seen that in this formulation the general problem is impossible to solve. For example, consider a genome of length 20 composed entirely of the character A. Assume that 8 random fragments of length 5 are sampled from the genome as shown in Fig. 9. The layout problem clearly has no unique solution—indeed any placement of the reads along the genome is possible. It can be argued that ultimately it is the DNA sequence of the original molecule that needs to be reconstructed, the exact placement of all the reads being irrelevant. This problem (the consensus problem) is, however, also impossible to solve in this case. The only information we can glean from the reads is that the genome is at least 5 base pairs long. Any string of As of at least 5 bases can be explained by the reads, indeed Fig. 9 shows some possible reconstructions. It is thus clear that in the general case additional information is required to solve the assembly problem. Please note that the theoretical example we chose is not entirely unrealistic. By replacing the character A with any other DNA string, the example becomes the problem of assembling **tandem repeats:** short stretches of DNA that occur in many copies, in tandem, throughout the genome. Such repeats are very common, for example in the telomeric and centromeric regions of many eukaryotic chromosomes [47].

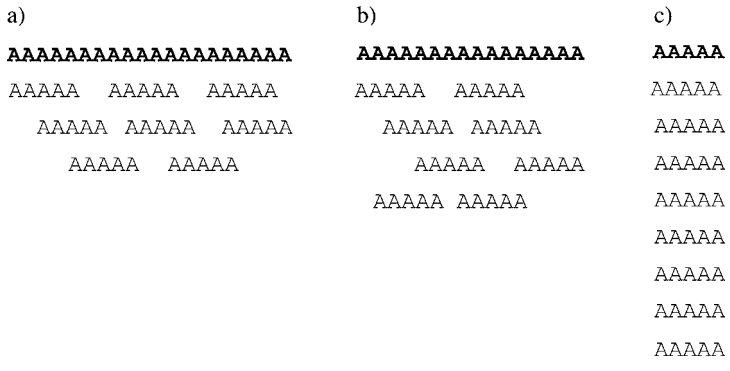


FIG. 9. Possible assemblies of the same set of 8 reads. The string in bold-type represents the supposed consensus sequence. a) represents the correct layout, b) and c) represent incorrect assemblies.

The assembly problem should thus be more generally defined as:

> Given a collection of reads with known DNA sequence together with **additional constraints** on their placement, find the DNA sequence of the original molecule.

The phrase "additional constraints" is intentionally vague as it encompasses a wide range of types of information. The constraints generally fall into three categories:

- **constraints on the global placement of the fragments**—this category refers to information that links a particular fragment or read to a specific place in the genome. A typical example is physical map data that assigns a fragment to a specific chromosome. This type of information is essential for the hierarchical sequencing strategy described in Section 3.4.
- **constraints on the relative placement of fragments or reads**—this category typically encompasses information that links together groups of reads, for example all the reads obtained by sequencing a same clone. The "overlap" constraint—the basis of any assembly program—is the most common constraint in this class. Another common type is the "mate-pair" relationship between the two reads sequenced from the opposite ends of a fragment. Clone walking techniques commonly used in finishing yield multiple reads that are all obtained by sequencing different sections of the same clone, and therefore must be held together during the assembly process.
- **statistical assumptions**—this class involves assumptions on the characteristics of the random process of shearing the DNA into fragments. Referring to the example at the beginning of this section, knowledge of the estimated length

of the genome, and the assumption that the fragments were selected through a uniform random process, allow us to favor the reconstruction in Fig. 9(a) over that in Fig. 9(c) and thus correctly resolve the repeat without the need to find the exact location of each read. Statistical methods allow an assembler to validate a particular layout by estimating the probability that the layout would occur in a random shotgun experiment. This approach was introduced by Myers [48] and is described in more detail in Section 4.2 in the context of repeat identification.

In practice, all the data provided to the assembler contain errors, making it difficult or impossible for the assembler to correctly identify the original DNA sequence. The assembly problem should thus be restated as having the goal to obtain a sequence that is "close" to the original sequence. There is, unfortunately, no universally accepted quality standard, though a widely used measure requires less than 1 difference in every 10,000 bases between the assembled sequence and the original DNA. This standard, commonly referred to as the **Bermuda standard**, was developed in the context of the sequencing of the human genome by an international consortium (for a summary of this standard see [49]).

3.1 Shortest Superstring

Initial attempts at understanding the assembly problem used a simplified theoretical model: **the shortest superstring problem**. Under this model, given a set of reads, the problem is to find the shortest string (the superstring) that contains all the original reads as substrings. Clearly a solution to this problem in the case of the reads described at the beginning of Section 3 would yield an incorrect assembly by reconstructing a genome that is 5 bases long (Fig. 9(c)). Furthermore this abstract problem ignores the presence of sequencing errors, inherent to all current sequencing technologies. Nonetheless an analysis of error and repeat free cases is an essential step in understanding the complexity of the general case.

In its general definition, the shortest superstring problem was proven to be NP-hard [50], and also MAX SNP-hard [51]. In other words the problem cannot be efficiently solved, neither can an approximate solution be found that produces a string arbitrarily close to the shortest superstring. Research was therefore devoted to defining the lowest approximation factor that can be efficiently computed. Most notably a simple greedy algorithm [52] was proven to yield a solution that is at most 4 times longer than the shortest superstring [51], though it is conjectured that the greedy algorithm comes within a factor of 2 of the optimal answer. A slightly modified greedy approach can be shown to come to within a factor of 3 to the optimal solution. Further results [53,54] proposed closer approximation schemes, though no approach yet reached the conjectured two-fold approximation factor.

The original greedy approach forms the basis of the first assembly algorithms and thus warrants a more detailed explanation. Its basic structure is:

1. add all the reads to a common store as singleton contigs (contigs with only one read)
2. find the two contigs with the best overlap, combine the contigs and add the resulting contig to the store
3. repeat the prior step until no more contigs can be joined.

This algorithm needs to be modified in order to be used in practice. Sequencing reads usually contain sequencing errors, inherent to all current sequencing technologies. Some amount of error is also introduced by the cloning technologies. In the presence of errors, the shortest superstring problem can be formulated as the task of finding the shortest sequence S that contains each read as an "approximate" sub-string, that is, each read r matches a substring of S with less than e errors, where e is an estimate of the error rate. Unsurprisingly, this problem was also shown to be NP-complete [55].

In the presence of errors, the definition of overlap from step 2 of the greedy algorithm needs to be modified to account for these errors. Two reads are said to **overlap** if they align such that the only differences between them can be explained by sequencing or cloning errors. The differences, or the edit-distance between the reads, are identified through a standard alignment algorithm (e.g., [56]) and only those alignments with low error rates are used. Assembly algorithms can typically tolerate alignments with between 2.5% and 6% insertion/deletion rates. Furthermore, the alignment between the sequences must be "proper", that is, the alignment must start and end at sequence ends. This requirement addresses spurious overlaps caused by repeats (see Fig. 10).

The redefinition of the notion of overlap requires us to reexamine the quality of an overlap. The "best" overlap is no longer the longest one. Practical implementations use one or more of the following measures when determining the quality of an overlap:

- Length of the overlap region;
- Number of differences, or conversely the percentage of bases shared by the two sequences as a fraction of the length of the overlap;
- Score of the alignment comprising four elements: reward for a good match, substitution penalty, gap opening penalty, and gap extension penalty;
- Quality adjusted score of the alignment—the four components of the score are adjusted to take into consideration the quality scores assigned by the base caller;
- Number of mate-pairs confirming (or conflicting with) the overlap.

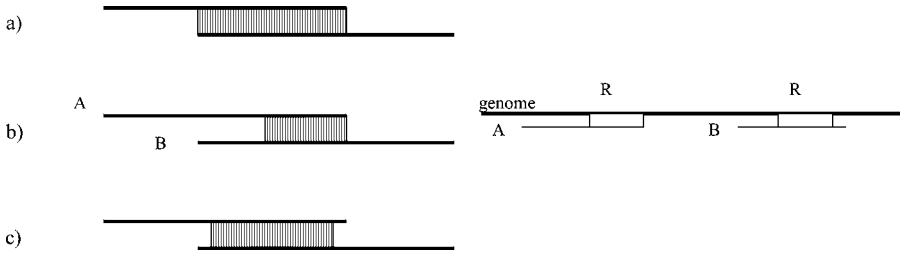


FIG. 10. Proper (a) and repeat induced (b) overlap between two reads (caused by repeat R). (c) represents a proper overlap in the case when errors are present in the reads. The unaligned regions are called "overhangs".

3.2 Overlap-Layout-Consensus

The original greedy approach to sequence assembly is inherently local in nature as only those contigs being merged are examined by the algorithm. Longer range interactions between reads can be considered, however this information is not easily incorporated in the standard algorithm, leading to complex implementations. Peltola et al. [57] and Kececioglu and Myers [48,58] introduced a new theoretical framework that addresses the global nature of the assembly problem: the **overlap-layout-consensus (OLC)** paradigm. They refine the problem by decomposing it into three distinct sub-problems:

- **overlap**—find all the overlaps between the reads that satisfy certain quality criteria.
- **layout**—given the set of overlap relationships between the reads, determine a consistent layout of the reads, i.e., find a consistent tiling of all the reads that preserves most of the overlap constraints.
- **consensus**—given a tiling of reads determined in the **layout** stage, determine the most likely DNA sequence (the consensus sequence) that can be explained by the tiling.

It is important to note that the greedy approach is, in some sense, an OLC algorithm, despite the fact that in many implementations the three distinct stages are not clearly delimited.

The main component of any OLC algorithm is the layout stage, originally formulated in graph theoretic terms. The overlap stage generates a graph whose nodes represent the reads. Two nodes are connected by an edge if the corresponding reads are involved in a proper overlap (as defined by preset quality criteria such as those discussed in Section 3.1). The layout problem can thus be defined as the task of finding an interval sub-graph that maximizes a particular target function. In other words,

the layout algorithm must find a sub-graph that contains only those edges representing overlaps consistent with a placement of the reads viewed as intervals along a line. As an optimization target, Peltola et al. [57] look for a layout that "uses best possible overlaps" while Kececioglu and Myers [58] attempt to maximize the weight of the resulting sub-graph, given a set of weights corresponding to the quality of the overlaps. They also show that, under this definition, the layout problem is NP-complete and propose a greedy approximation algorithm as well as a method for enumerating a collection of alternative solutions. Myers [48] further introduces a variant of this problem that generates the layout that best matches the statistical characteristics of the fragment shearing process under Kolmogorov–Smirnov statistics.

Layout algorithms operate on huge graphs (from tens of thousands of nodes in the case of typical bacterial genomes to tens of millions in mammalian-sized genomes) making the global optimization of the layout practically impossible. Practical implementations attempt to solve the layout problem in a greedy fashion, by starting with those regions of the graph that can be resolved unambiguously. Thus Myers [48] proposes a series of transformations that convert the initial overlap graph into a **chunk graph**, where a chunk is a maximal interval sub-graph. In other words, chunks represent sections of the genome that can be unambiguously resolved, i.e., the sections between repeats. Repeats cause branches in the overlap graph (see Fig. 11) and signal the end of a chunk. The complexity of the graph is thus greatly reduced allowing the use of more sophisticated algorithms in order to obtain a consistent layout of the chunks. Usually at this stage the chunk graph is augmented with additional information, such as mate-pair data.

Variants of this idea were used in practical implementations, for example Celera Assembler [8] (where the chunks are called **unitigs:** uniquely assembleable contigs) and Arachne [18] start the assembly process by identifying the unambiguous sections

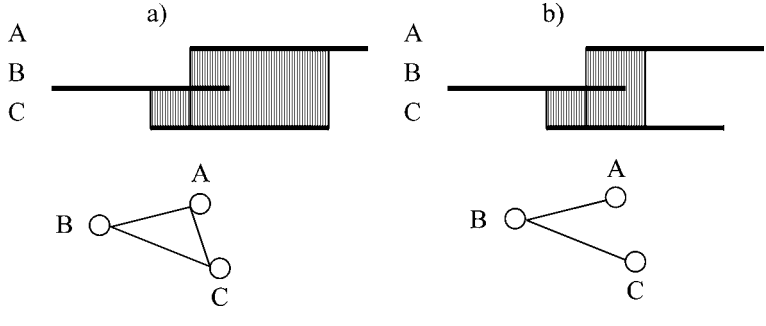


FIG. 11. Effect of repeats on overlap graph. a) represents the overlap of three reads in the absence of repeats. The overlap graph contains an edge for each pair of reads. b) represents the overlap between three reads at the boundary of a repeat. The overlap graph lacks the edge between A and C.

FIG. 12. Paired-pair: two fragments whose end-reads overlap.

of the genome then use a scaffolding step to generate the final layout. Arachne uses the mate-pair information from the beginning of the assembly process by identifying **paired-pairs**, that is, pairs of shotgun fragments of similar lengths whose end sequences overlap (see Fig. 12). Paired-pairs represent a high confidence structure that can be used to seed the contig building process. In contrast, Celera Assembler uses mate-pair information in the later stages of assembly.

3.3 Sequencing by Hybridization

Idury and Waterman [59] proposed an alternative approach to sequence assembly, that shifts the focus of the algorithm from the reads to the overlaps between the reads. They break up the shotgun reads to emulate a virtual **sequencing by hybridization** (SBH) experiment, and then attempt to solve this latter problem. The sequencing by hybridization technique involves building an array (in the form of a micro-chip) of all possible ***k*-tuple** probes. The DNA being sequenced is **hybridized**[10] to the chip in order to find all the k-tuples present in the DNA sample. The SBH problem can thus be formulated as: find the DNA strand whose ***k*-tuple spectrum** (set of all possible k-tuples contained in the DNA) is the observed set of k-tuples. Note that the problem is different from a standard shotgun sequencing problem with reads of length k, since the tuples represent a uniform sampling of the genome (see Fig. 13). Furthermore, the information provided is simply the presence or absence of a particular k-tuple and not the multiplicity of the tuple in the genome. In shotgun sequencing it is often easy to detect repetitive sections of the genome by identifying DNA strings over-represented in the read set.

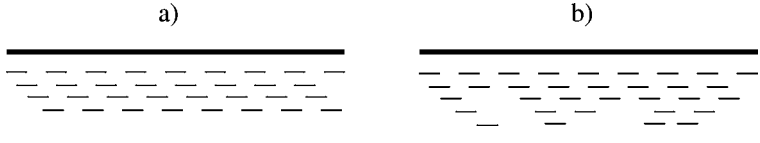


FIG. 13. a) represents the uniform coverage of a genome with k-mers. b) represents the (uneven) coverage by a randomly sheared set of fragments.

[10]**hybridization**—process through which a short single stranded nucleotide string attaches to the complementary string in the DNA strand being analyzed.

The SBH problem can be represented in graph theoretic terms as follows: given a set of k-tuples, we construct a graph whose nodes represent all the $(k-1)$-tuples in the set. Two nodes are connected by an edge if the corresponding $(k-1)$-tuples represent the prefix and the suffix of one of the original k-tuples. Thus the k-tuples are implicitly represented by edges in the SBH graph. A solution to the SBH problem is represented by a path through the graph that visits every single edge (k-tuple). This problem is related to a classic problem in graph theory—the Eulerian path problem—that requires finding a path through the graph that uses every single edge exactly once. Note that in the case of SBH, a particular k-tuple may be used multiple times if it occurs in a repeat.

The Eulerian path problem is generally easy to solve—if such a path exists it can be found in linear time with respect to the number of edges in the graph. The instances of the problem induced by SBH experiments, however, limit the applicability of this approach to sequencing short pieces of DNA. On the one hand, the k-tuple array must contain all possible k-tuples, thus physically limiting the size of k, and implicitly the size of strings that can be sequenced by SBH. As an example, a 2-tuple array containing the 16 possible di-mers cannot be used to sequence any DNA string longer than 17 bases. On the other hand, hybridization errors and repeats contained in the DNA string complicate the graph. While finding an Eulerian path is easy, the task of finding the correct path—the path corresponding to the original DNA molecule—is a much harder problem. For example, the graph in Fig. 14 can explain two different DNA strings corresponding to reorganizations around repeat R. The graph alone does not contain the information necessary to make the correct choice. Note however that the repeat is immediately recognizable in the graph. Figure 15 contains the example of a tandem repeat and the corresponding structure in the overlap and SBH graphs. The repeat can be easily identified in the SBH graph, however it is not immediately obvious in the overlap graph.

Despite the limited applicability of the SBH technique to the actual sequencing of DNA, its theoretical structure leads to an alternative approach to shotgun sequence

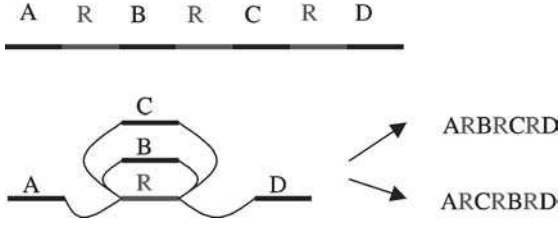


FIG. 14. SBH graph for a 3-copy repeat R. The graph supports two different reconstructions of the genome: ARBRCRD and ARCRBRD.

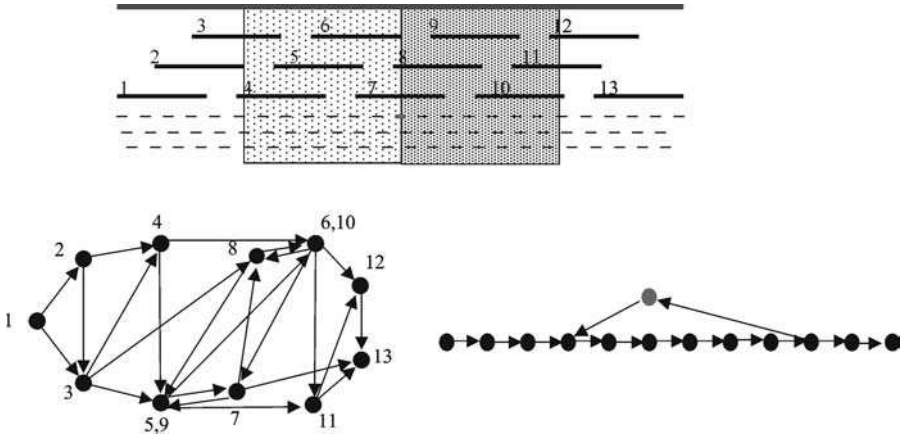


FIG. 15. Tandem repeat (shaded regions in the top picture) and its representation in the overlap graph (bottom left) and the SBH graph (bottom right). The numbered lines in the top region represent reads while the short segments correspond to k-mers. The SBH graph does not contain a representation of all the k-mers due to lack of space. The k-mer represented in gray spans the boundary between the two copies of the repeat and is therefore unique in the genome. The loop in the graph corresponds to those k-mers contained in the repeat region.

assembly. Idury and Waterman [59] proposed using the shotgun reads to simulate an SBH experiment. They break up each read into overlapping k-mers. The combined k-mer spectra of all the reads correspond to the k-mer spectrum of the original DNA, and thus solving the SBH problem is equivalent to solving the initial shotgun sequence assembly problem. It is important to note that the *in-silico* SBH experiment does not impose limitations on the size of k. The algorithms need only process those k-mers actually present in the read set. For a genome of size G, we expect $G - k + 1$ such k-mers, a number that is generally much smaller than 4^k (the set of all possible k-mers). The technique is thus, at least theoretically, applicable to arbitrarily sized genomes. The authors notice, however, that sequencing errors lead to spurious k-mers that greatly complicate the graph.

Idury and Waterman's mainly theoretical work was extended by Pevzner et al. [60–62] leading to a practical implementation, a software package called Euler. They addressed several issues of practical nature. First of all, their algorithms depend on an error correction module, since sequencing errors hopelessly tangle the Euler graph. A description of this module is provided in Section 4.2. Secondly, they use the initial reads as a guide in generating the Eulerian path (this idea was introduced by Idury and Waterman), leading to a new problem—the Eulerian superpath problem: find an Eulerian path that confirms most reads in the input. The constraints provided by the reads help resolve most short repeats. Thirdly, the authors use the mate-pair

information to guide the algorithm in an attempt to resolve longer repeats. Finally, the remaining repeats that cannot be resolved are identified and reported. This is an important feature of the program as it allows the users to design further experiments to resolve the correct structure of the genome [46].

3.4 Hierarchical Assembly

Before the success of the *Drosophila* sequencing project [7], the assembly of whole genome shotgun data was considered impossible for any genomes larger than bacteria (6–8 Mbp). At that time, the most widely used assembly program was phrap, which was easily confused by repeats (see, for example, analyses presented in [60, 63,64]), an ubiquitous feature of eukaryotic genomes. Other commonly used assemblers, such as TIGR Assembler or CAP, required large amounts of memory and were therefore unable to assemble large data-sets. In order to overcome the limitations of the available software, scientists followed a hierarchical approach to sequencing large organisms. They started by breaking up the original DNA into a collection of large fragments, with sizes between 50 and 200 kbp. Most commonly, these fragments were cloned into a **Bacterial Artificial Chromosome (BAC)** vector, though some projects used **fosmids** or **cosmids**. These large DNA fragments offered several advantages. Firstly, their small size allowed the use of existing assembly programs. Secondly, they contained few internal repeats. Only those repeats that appear in multiple copies within the same fragment would confuse the assemblers. The small size of the fragments guarantees that certain classes of repeats, specifically repeats whose copies are spatially separated along the genome, would not pose assembly problems. Thirdly, the fragments were long enough to contain sufficient physical markers that allow their unambiguous placement along the genome.

The hierarchical sequencing approach proceeds in three steps. In the first step, scientists map each of the fragments to a unique location along the genome. This map allows them to identify a **minimal tiling path**, that is, a collection of overlapping fragments that cover the entire genome. This map is minimal in the sense that we need to choose such a set of fragments that have minimal overlap with adjacent fragments. Since each fragment will be individually sequenced, the regions of overlap will end up being sequenced twice, leading to an increased cost for the procedure. At the same time the overlaps need to be large enough to allow the researchers to paste the genome back together. In the second step of this sequencing technique, each of the individual fragments is sequenced through the shotgun method. Finally, the finished fragments are assembled together using the overlapping regions as a guide. Note that the overlaps between fragments are essential in generating the correct sequence of the genome since the initial fragment map is inherently imprecise due to the low resolution of common physical mapping techniques.

This hierarchical method thus introduces the need for two specialized assembly programs: one that performs the individual assemblies within each fragment, and one that pastes together the finished fragments using the overlaps and the initial fragment map as a guide. For the first task existing assembly program such as phrap and TIGR Assembler are commonly used, as the small size of the BACs does not impose significant assembly challenges (though complex repeats remain a problem even in such a localized context). The latter task is quite easy and, typically, the final assembly step is performed either manually or through a collection of simple computer programs. This was the approach used in sequencing the model plant organism *Arabidopsis thaliana* [65]. The initial assembly of the human genome by the Human Genome Sequencing Consortium, however, required a much more sophisticated approach, given the quality of the available data. A special program, called GigAssembler [66], was used to combine a collection of finished and partially finished BACs, as well as many individual contigs. The problems were compounded by errors in the physical mapping data and mis-assemblies of the contigs or BAC sequences. GigAssembler uses techniques similar to those developed for shotgun-sequence assembly, therefore some of the technical details of the implementation will be discussed later in that specific context.

The hierarchical sequencing approach led to active research in the development of specialized techniques for obtaining the initial BAC map. Such research addresses both the laboratory technologies involved in physical mapping [67] as well as the software issues involved in generating and analyzing such maps.

Researchers at the Baylor College of Medicine developed a BAC mapping technique that combines the cost advantages of shotgun sequencing with the simpler algorithms required by the hierarchical method. They follow a hybrid approach wherein the mapping of the BACs along the genome is replaced by a low coverage "light" shotgun of a collection of BAC clones. At the same time, the genome is sequenced using the standard shotgun sequencing technique. The last step of their technique involves mapping the shotgun reads to individual BACs, using the reads generated through the light shotgun of the BACs as anchors. The BACs thus serve as a guide to clustering the shotgun reads into more manageable blocks. This technique represents the basis of the assembly program Atlas. Please note that, in the absence of an actual BAC map, the final step of joining the individual BAC assemblies together becomes considerably more difficult. Similar BAC "recruiting" techniques form the basis of the Phusion [20] and RePS [68] assemblers and was also proposed as an alternative to whole-genome-shotgun in the assembly of the human genome at Celera [69].

Cai et al. [28] propose a refinement of this hybrid technique, called **Clone Array Pooled Shotgun Strategy (CAPSS)**. They place each BACs DNA within the cells of a two-dimensional matrix, then pool the DNA within each row and column of the

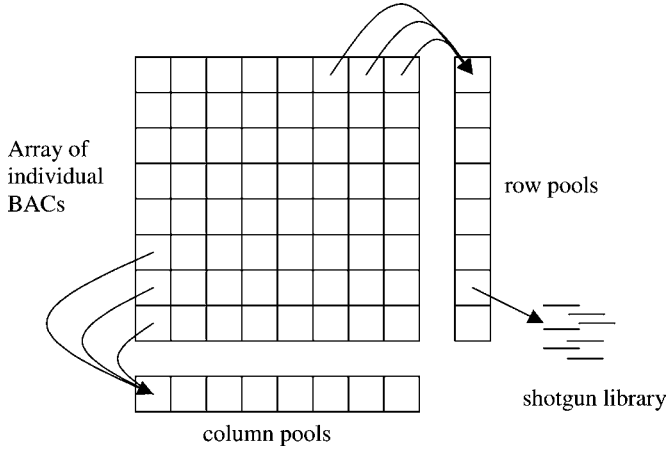


FIG. 16. Clone-Array Pooled Shotgun Strategy.

matrix as shown in Fig. 16. The light shotgun step is then applied to each pool, thus reducing the number of libraries created. In the case of a collection of n BACs, the initial approach requires the generation of n libraries, while the pooled DNA method requires only $2\sqrt{n}$ libraries. Each BAC will thus be represented in the reads from two different libraries, one from the row, and the other from the column containing the BAC's well. In the last step of CAPSS, the shotgun reads are assembled together and the resulting contigs used to identify the correct mapping of reads to BACs. The contigs that contain reads from both column i and row j of the CAPSS array correspond to reads generated from the BAC clone in well (i, j). Furthermore, this strategy can be extended to also produce a map of the BAC clones, thus circumventing the need for an additional mapping step. This technique, called **Pooled Genomic Indexing (PGI)** [70] requires two separate arrays for each set of BACs. The placement of BACs in the wells is shuffled between the two arrays so that no two clones occur within the same row or column in both arrays. As a result, the deconvolution of the contigs also yields the relative placement of pairs of BACs, information that is sufficient to generate a map of the genome.

It is important to note that hierarchical approaches are also very important in whole genome shotgun sequencing. Indeed, such approaches are essential during the finishing stages of a sequencing project. As an example, the following hierarchical approach is commonly used to correctly assemble repeats. Clearly repeats are only a problem if reads corresponding to two or more nearly-identical copies of a repeat are being assembled at the same time. When faced with a potentially mis-assembled repeat, researchers attempt to identify fragments whose ends are anchored in the unique

areas flanking a particular repeat copy. Note the importance of having libraries of multiple sizes as they allow the resolution of different classes of repeats. The fragments are then further sequenced, either through directed sequencing or through a separate shotgun experiment (depending on size). The resulting reads can be safely assembled together since they represent a single copy of the repeat. The resulting contig, together with the flanking unique sequence can then be used as a building block in assembling the rest of the genome, without the risk of mis-assembly. It is important for the assembly program to allow such a hierarchical approach by providing a means for "jump-starting" the assembly with the already generated contigs.

3.5 Machine Learning

A discussion of general assembly paradigms would be incomplete without a brief discussion of approaches based on machine learning techniques [71–73]. In fact an early approach to the shortest superstring problem phrased it in machine learning terms as the problem of learning a string from a collection of random substrings [74].

Parsons et al. [71] explore the applicability of genetic algorithms to the sequence assembly problem. Genetic algorithms attempt to optimize a target function, in this case a measure of the errors in the tiling of reads, by maintaining a "population" of candidate solutions. The population is allowed to "evolve" towards the optimum through a set of operators (mutation, crossover) imitating the evolution of biological systems. Their work can best be described as a "proof of concept" as the largest data-set they analyze contains 177 fragments that cover 34 kbp. They report that the genetic algorithm is able to out-perform a simple greedy approach for such data-sets.

A more interesting idea is introduced by Goldberg and Lim [72,73] who examine a different approach to using machine learning techniques. They correctly notice that assembly algorithms consist of a complex set of inter-connecting modules controlled by a large number of parameters. They propose the use of a "generic" assembly algorithm that can learn the best mixture of parameters from a small set of examples (sequences whose correct assembly is known).

4. Assembly Modules

4.1 Overlap Detection

The basic assumption of shotgun sequencing is that sequence similarity between two reads is an indication that the reads originate from the same section of the genome. All assembly algorithms must therefore identify similarities between reads.

The specific algorithmic approaches to the task have evolved throughout the years, as increasingly more complex sequencing projects were tackled through the shotgun method. The earliest algorithms involved either iteratively aligning each read to an already generated consensus [2] or comparing all the reads against each other [57]. Most recently the detection of read overlaps involves sophisticated techniques meant to reduce the number of pairs of reads being analyzed. For example in the case of the human genome, a full pair wise comparison of all 50 million reads from a $5\times$ shotgun sequencing experiment would be prohibitive, especially as, at least theoretically, each read overlaps only a small number of other reads (approximately 5 other reads). Furthermore, recent algorithms based on the sequencing-by-hybridization paradigm [59,62], avoid the explicit computation of read overlaps since these are implicitly represented in the graph constructed by such algorithms.

The overlaps implied by sequence similarity between reads fall in two classes: "real" overlaps—the reads were obtained from the same region of the genome; "repeat-induced" overlaps—the reads belong to two distinct repetitive regions of the genome (see Fig. 10). Ideally, an assembler should only use the "real" overlaps since "repeat-induced" overlaps lead to ambiguities in the placement of the reads. Such ambiguities are often hard or impossible to resolve and most assemblers use several heuristics to reduce the number of repeat-induced overlaps generated.

Two reads are said to overlap only if the overlap is *proper*, that is either one read is entirely contained in the other, or the two reads properly **dove-tail** as shown in Fig. 10(a). This heuristic avoids considering short repeats by eliminating the overlaps represented in Fig. 10(b). The unaligned regions in this figure are called **overhangs**. A second heuristic requires the reads to be highly similar in the region of overlap. In the absence of sequencing errors, two reads that were generated from the same region of the genome will have identical sequences, thus any "imperfect" alignment would indicate an overlap induced by a repeat whose copies have diverged during evolution. In practice, however, sequencing errors must be taken into account, and therefore assembly algorithms must tolerate imperfect alignments. One or more of the following "imperfections" are usually allowed when considering overlaps: base substitutions, base insertions or deletions, and overhangs. Please note the delicate balance between the sequencing error tolerated by an assembler, and the ability to detect repeat induced overlaps. An algorithm that requires perfect overlaps between reads would only be confused by exact repeats (identical stretches of DNA that occur in multiple places throughout the genome). Such an algorithm will, however, only identify a small percentage of all true overlaps due to sequencing errors. An algorithm tolerating a 3% sequencing error rate (this number corresponds to the estimated error rates at the large sequencing centers) will identify most true overlaps. At the same time the algorithm will identify more repeat-induced overlaps, specifically those due to repeats that have less than 3% differences between repeat instances.

Unfortunately, many classes of repeats exhibit rates of divergence between copies less than 3% [75]. This correlation between sequencing error rates and the ability to detect repeat-induced overlaps led to the development of error correction algorithms which will be discussed in the next section. The removal of sequencing errors allows the overlap algorithm to be more stringent, thereby reducing the number of repeat-induced overlaps without a decrease in the capacity to identify true overlaps.

In the absence of errors, the overlap problem can be solved in an efficient manner, for example through the use of suffix trees [76,77] or suffix arrays [78,79]. Suffix trees [80] and arrays [81] are data-structures designed to efficiently store all the suffixes of a particular string. Both data-structures are space-efficient, requiring an amount of space proportional to the total size of the string being stored. Suffix arrays are three to five times more space-efficient [81] than suffix trees, thus being more appealing when handling large genomes. Furthermore suffix trees can be built in linear time [80,82,83] while suffix arrays can be built in expected linear time [81]. It is easy to understand how suffix trees and arrays can be used to solve the overlap problem. A correct dove-tail overlap implies that a prefix of read A is identical to a suffix of read B. In order to identify all reads that overlap a particular read r, it is sufficient to identify the reads whose suffixes match a prefix of r. Obtaining all overlaps simply involves querying a database of suffixes with each read in turn, an operation that can be done in a time- and space-efficient manner. Kosaraju and Delcher [76,77] propose an elegant extension of this idea, leading to a linear time implementation of the basic greedy algorithm through the traversal of an augmented suffix tree.

Practical implementations must, however, tolerate sequencing errors and therefore the algorithms involved are significantly more complex. The overlap between two reads is generally computed through variations of one of the standard string alignment algorithms [56,84]. Such algorithms usually require quadratic space and time, which represents a considerable computational burden even though the strings involved are short (reads are generally shorter than 1000 base pairs). The space requirement can be easily reduced to a size proportional to that of the input sequences (see, for example, [85]). In the case of shotgun sequence assembly, the difference between overlapping fragments is relatively small, corresponding to the small sequencing-error rates (less than 3%) typical of current sequencing technologies. This observation led to the introduction of bounded-error alignment algorithms (e.g., [86]) whose running time depends on the tolerated error rate. Such algorithms run in time proportional to $\varepsilon \cdot n$ where ε is the maximum error rate allowed in the alignment and n is the length of a sequencing read. The quadratic running time of standard alignment algorithms stems from the use of a dynamic programming approach that utilizes a two dimensional alignment matrix. When the error rate is bounded, an alignment algorithm need only examine a small band (of size proportional to the error rate) around the diagonal of this matrix, leading to the afore-mentioned speed improve-

ment. Such algorithms are commonly known as **banded alignment algorithms**. For a more in-depth description of the various string alignment algorithms the reader is referred to [87].

Besides identifying the specific overlap of two given reads, an overlap algorithm must also determine which pairs of reads overlap. This step of the algorithm can also be more efficiently implemented when the error rates are small. In the case of unbounded error rates an overlapper must examine all possible pairs of reads, leading to a quadratic number of pair wise comparisons—an inherently inefficient process. The following observation leads to a more efficient approach in the case when the number of errors tolerated by the algorithm is small. A low number of errors implies that two overlapping reads must share several identical stretches of DNA. As described above, algorithms that identify exact matches are very efficient leading to a two-step process for identifying read overlaps. First, a set of short identical matches between the reads is identified, then only those pairs of reads that share the same set of exact matches are considered in more detail. Chen and Skiena [78] estimate that this simple heuristic reduces by a factor of 1000 the number of pairs of reads that need to be considered. Furthermore, the exact matches between two reads can be used to "seed" their alignment, greatly reducing the amount of time required to perform a detailed alignment. Such techniques are commonly used to speed up database search algorithms such as BLAST [88] or FASTA [89]. The AMASS assembler [90] further extends this approach by entirely skipping the detailed alignment step. Thus read overlaps are identified by examining the pattern of shared exact matches, a detailed alignment being postponed until the last phase of the assembly—the generation of the final consensus.

Besides the suffix-tree and suffix-array techniques described above, the detection of exact matches between pairs of reads is usually performed by building a map (usually under the form of a hash table) of all k-mers present in the reads, keeping track of the set of reads that contain each k-mer. A simple pass through the table is sufficient to identify all pairs of reads that have a particular k-mer in common. Variants of this simple approach are used by virtually all assemblers used in practice [5,8,18, 20–22,64,66,91,92]. Tammi et al. [93] suggest an extension of this basic approach by structuring the k-mer database in such a way as to allow querying for inexact word matches. Specifically, they describe an method for finding all k-mers that have less than d differences from a given query k-mer, where d is a parameter corresponding to the expected sequencing error rate. Using this technique they hope to improve the sensitivity of the overlap stage of the assembly.

The choice of the length parameter k affects the sensitivity of the overlap detection algorithm. Short k-mers occur frequently in the DNA sequence, leading to the identification of many potential read pairs that need to be evaluated by the algorithm. The use of long k-mers may cause the algorithm to miss many true overlaps due to

the effect of sequencing errors. The k-mer size generally ranges between 10 bp (as in GigAssembler [66]) and 32 bp (as in TIGR Assembler [5]), the specific choice depending on the nature of the data processed by the assembler. The GigAssembler was designed to tolerate the large error rates inherent to the heterogeneous nature of the Human Genome Project, while TIGR Assembler could afford a more stringent value due to the high quality of the data generated by shotgun sequencing of bacteria.

In general, assembly programs identify all distinct k-mers present in the reads. Roberts et al. [94] note that it is sufficient to store only a sub-set of all k-mers therefore significantly reducing the time and space requirements of the overlap routine. For each set of m consecutive k-mers—consecutive means each k-mer is shifted by one base from the previous one—they only store the **minimizer**, i.e., the smallest k-mer in terms of a specific lexicographic order. They show that any reads that share an exact match of more than $m + k - 1$ bases must have at least one such minimizer in common. For appropriately chosen values of m and k, the minimizer technique greatly reduces the complexity of the overlap stage without missing true overlaps. The authors also describe a procedure based on file sorting that allows them to trade off expensive RAM memory for much cheaper disk space, without a significant degradation in performance.

The overlap stage of an assembler trivially lends itself to parallelization. A set of n reads can be partitioned into K sub-sets. This leads to K^2 distinct overlap tasks that can be performed in parallel, corresponding to all possible pairings of the K sets. The overlap task pairing sets i and j leads to the identification of all reads in set i that overlap reads in set j. This approach was used at Celera in order to take advantage of their large processor farm [8]. Note, however, that their approach converts a task that can be solved in linear time into a quadratic process. Their technique can, therefore, only provide an advantage over the single processor solution for small values of K. Parallelization of the overlap stage was also proposed by Huang et al. [95] as a main component of their assembler (PCAP) specifically designed to handle mammalian-sized genomes.

The overlaps identified in this stage of assembly provide the input to the layout stage. They are, however, also used to identify specific features of the genome: chimeric reads, missed overlaps, repeats, and sequencing errors.

Chimeric reads (see Fig. 17) are an artifact of the sequencing process wherein two distinct sections of the genome are represented in the same read. Such errors are ubiquitous in gel-based sequencers though they have become much less common since capillary-based sequencers have been introduced. They can also be an artifact of the cloning process, due to the recombination of the DNA fragments. Since chimeric reads do not represent any section of the genome, their overlaps with other reads can be used to detect the “separation point” that is the place in the read where the two distinct sections of the genome come together (see Fig. 18). Although rare,

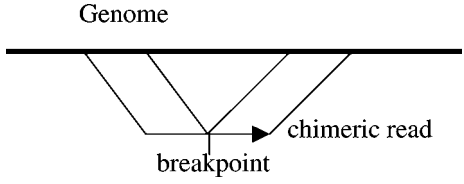


FIG. 17. Chimeric read. The read contains DNA from two unrelated sections of the genome.

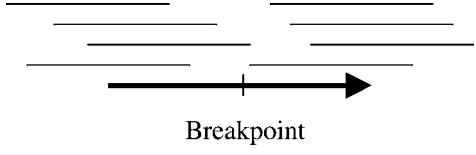


FIG. 18. Identification of chimeric read from overlap with other reads. The breakpoint is not spanned by any other read in the genome.

chimeric reads can confuse the assembler therefore some assembly packages include a module that detects and eliminates potential chimeras [5,18,64,92].

Chen and Skiena [78] also propose a method for identifying those overlaps that might have been missed by a stringent overlap algorithm. They identify the transitive relationship between three reads where only two of the overlaps had been identified (see Fig. 11(b)). Such a situation, commonly induced by repeats, can also be caused by sequencing errors, therefore it is useful to attempt to identify such overlaps missed by a stringent overlap algorithm. Finally, the overlaps between reads can be used to identify and correct sequencing errors, and to detect repetitive regions that might confuse the layout stage. These problems will be discussed in detail in the next section.

4.2 Error Correction and Repeat Separation

The previous section described the interplay between the tolerance of an assembly algorithm to sequencing errors and its ability to correctly identify and ignore repeat-induced overlaps. Repeats are probably the biggest challenge to shotgun sequence assembly inasmuch as some argued that their presence would make impossible a whole-genome shotgun approach to sequencing the human genome [6]. Sequencing errors limit the ability of an assembler to detect and correctly resolve such repeats. Furthermore, their presence leads to more complex graph structures that need to be handled during the layout phase of assembly algorithms. For example, the Eulerian

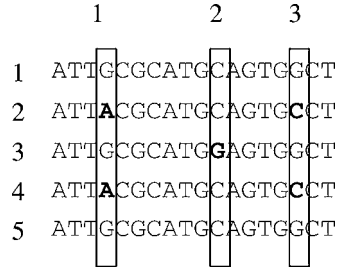


FIG. 19. Correlated "errors" between reads can be evidence of mis-assembled repeats. Columns 1 and 3 represent such evidence. The disagreement in column 2 is most likely an error as it occurs in only one of the reads.

path approach of Pevzner et al. [60] is affected by sequencing errors to such an extent that their algorithms depend on an error correction module.

It is therefore not surprising that many of the recent developments in the field of sequence assembly address specifically the task of automatically correcting sequencing errors during a pre-processing stage of assembly. The basic idea behind all error correction approaches is statistical in nature. It assumes that sequencing errors occur in a random fashion within each read, furthermore the distributions of errors in distinct reads are independent of each other. The probability that two overlapping reads would contain the same sequencing error at the same exact location is therefore practically negligible. Within a tiling of reads corresponding to a specific section of the genome an error at any particular position would occur in only one of the reads (see Fig. 19). Correlated errors between reads represent strong evidence of either the presence of multiple distinct copies of a repeat, or the existence of multiple divergent **haplotypes**[11] in the DNA being sequenced. Please note that this discussion only applies to sequencing errors which can, in practice, be considered as the outcome of a random process. Other types of errors occur in shotgun sequencing projects, which do not have the same random behavior as sequencing errors. For example, the presence of long stretches of a same repeated nucleotide causes the sequencing reaction to "slip" leading to errors in all the reads containing the particular sub-sequence.

The most commonly used sequencing technique involves the identification of fluorescently tagged DNA as it passes in front of a detection mechanism. The physical output of a sequencer consists in four signals corresponding to the four different nucleotides (see Fig. 20). Specialized programs (called **base-callers**) use signal-processing techniques to identify the individual bases composing the DNA strand

[11]**haplotype**—Eukaryotic genomes generally contain two copies of each chromosome. Each copy is obtained from one of the parents, thus the two copies may differ from each other. Each of the alternative forms of the genotype (complement of genes) corresponding to the two chromosomes is called haplotype.

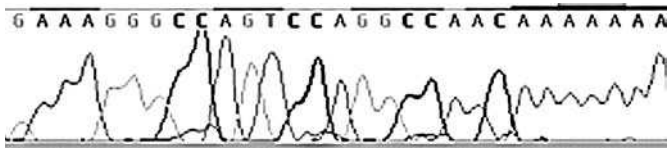


FIG. 20. The four signals, corresponding to each of the four bases, produced by an automated sequencer. This diagram is called a chromatogram.

being sequenced [96,97]. These programs also produce an estimate of the quality of each base, in terms of the log-probability that the particular base is incorrect ($qv = -10\log(p_{\text{error}})$). Such error estimates have been shown to be relatively accurate [97].

The first attempts at reducing the effects of sequencing errors used these error-rate estimates when computing read overlaps, thus allowing mis-matches if one or both of the bases had low qualities, and penalizing mis-matches between high-quality bases. Most of the early assembly programs [5,64,79,91,92] used this approach and the simple idea continues to be used in some of the recently developed assemblers [18,22]. Also note that base quality estimates are used by most assembly programs to roughly identify the high quality portion of each read (called **clear range**) since sequencing errors are significantly biased towards the ends of each read. Some assemblers (e.g., phrap [91]) perform this step internally, while others require specialized "trimming" software [98] to remove the poor quality ends of the reads.

Huang [92] was, to our knowledge, first to introduce the idea of using the alignment of multiple reads to identify the location of sequencing errors. For each read r he introduces the notion of an **error rate vector** which, for each base b, stores the largest error rate in a section of r bases starting at b as defined by alignments of read r with all the other reads it overlaps. The error rate vectors are then used to evaluate the overlaps between reads in order to identify chimeric fragments and repeat-induced overlaps.

Huang [92] and Kececioglu and Yu [99] were first to address the error correction problem in the context of a multiple alignment of a collection of mutually overlapping reads. They actually solve a complementary problem, that of separating multiple non-identical copies of a same repeat using correlated mismatches between the reads of the multiple alignment. The authors attempt to identify columns in the multiple alignment where reads disagree. If the disagreements between reads are correlated, i.e., occur in more than one single read, they refer to the columns as **separating columns** (columns 1 and 3 in Fig. 19). The assumption is that if a particular "error" occurs in a single read it is due to a sequencing error, however, multiple correlated "errors" indicate the collapse of two or more repeat copies. The assembly algorithm can thus avoid mis-assembling the repeat by removing the overlaps between reads

with conflicting bases in the separating column. A by-product of this approach is that disagreements between reads that do not occur in separating columns can be corrected as they most likely represent sequencing errors. Kececioglu and Yu attempt to reduce the number of false positives (columns incorrectly labeled as separating) by requiring multiple separating columns to confirm each other. Two columns are said to confirm each other if the corresponding mismatches occur in the same reads. They further propose a model for computing the probability that a particular set of c separating columns represent a false-positive event. Their model takes into account the sequencing error rate and the difference between repeat copies. Thus they provide a mechanism for identifying the number c of correlated separating columns necessary for separating repeats with a desired confidence level. The problem of correctly separating two repeat copies is easy if all separating columns support each other. In the case when the separating columns for a set of reads are not mutually correlated the authors introduce the **partition problem**: given a set of reads that disagree at defined separating columns, identify a partition of the reads into k classes such that the number of errors (differences between reads that belong to the same class) is minimized. The partition problem is equivalent to the ***k*-star problem**, a known NP-complete graph problem. The k-star problem requires finding a partition of the vertices of a graph into k stars such that the sum of the weights of the edges is minimized. They propose an optimal branch-and-bound solution to this problem. Myers [100] examines a similar problem for the case when there are two repeat copies ($k = 2$). This seemingly simpler problem is also NP-hard, thus he suggests a branch-and-bound algorithm together with a set of lower bound functions that are used to prune the search tree and greatly speed the algorithm.

A similar approach to repeat separation and error correction was used by Tammi et al. [101]. The main contribution of their work is the development of a statistical framework for computing the significance of a separating column (which they call defined nucleotide positions—DNPs) that takes into account the quality values of the underlying reads. They further require that each DNP be supported by another one in order to reduce the number of false positive. The authors show that their approach can greatly reduce the amount of misassemblies due to repeats by comparing their algorithm with the commonly used assembler `phrap` [102].

Roberts et al. [94] use a method that bypasses the need to compute a multiple alignment of all overlapping reads. Thus they notice that instead of correctly separating a set of reads into classes corresponding to distinct repeat copies it is sufficient to eliminate the overlaps between reads that appear to belong to different repeat copies. Once all such repeat-induced overlaps have been removed the assembler will correctly assemble them together into separate repeat copies. The authors use a heuristic rule called the **4–3 rule**, that examines overlapping sets of 4 reads and 3 separating columns.

Most assembly algorithms [5,8,18,20–22,64,68,90–92] use a simple heuristic to reduce the effect of identical repeats with large copy numbers. As described in the previous section, the overlaps between reads are usually computed by creating a map of all k-mers present in the reads. At this stage, the assemblers remove from consideration those k-mers that occur too often in the data since they most likely correspond to short repeats with a large number of instances.

All these error correction and repeat separation algorithms can be performed right after the overlap stage of an assembler, without requiring any larger structures such as contigs or scaffolds. In fact error correction can be performed even before overlaps are computed as shown by Pevzner et al. [60]. In the context of their Eulerian path approach, the overlaps between reads are not computed, rather they are implicitly represented in the SBH graph structure. The authors analyze the effect of sequencing errors on the k-mer spectrum of a reads. The spectrum of a string represents the set of all k-mers belonging to the read. For a set of reads they identify a set of "solid" k-mers, specifically those that belong to more than a pre-determined number of reads. In this context error correction can be performed by solving the **spectral alignment problem** [103]: given a string s and a spectrum T (in our case the set of solid k-mers) find the minimum number of mutations in s such that the spectrum of the resulting string is contained in T. The authors also propose a simple heuristic approach that does not require knowledge of the set of solid k-mers and takes advantage of the fact that errors in each read are relatively rare. Thus they formulate the **error correction problem**: given a set of strings S and a threshold Δ, find a set of fewer than Δ corrections in each read such that the number of k-mers in the spectrum of S is minimized. The rationale behind this approach is that each sequencing error leads to k erroneous k-mers therefore increasing the size of the spectrum of S. Removing a sequencing error would thus result in a reduction of the size of the spectrum of S by k words. Changing a correct base would not decrease the size of the spectrum as multiple reads contain the same base. The authors estimate that their approach eliminates more than 86% of all sequencing errors, though, on occasion, the method also introduces new errors by incorrectly changing correct bases. An attempt at validating this algorithm was reported by Tammi et al. [93] in the context of a comparison with their own error correction algorithm.

4.3 Repeat Identification

All methods described so far require that distinct copies of a repeat differ among each other as the algorithms rely on these differences to separate out the repeat copies. In the case of exact repeats most assemblers (with the notable exception of the Euler program [60]) choose to simply remove all overlaps between the reads belonging to the repeats. Thus the assemblers lose the ability to assemble the repeats,

however they gain a considerable reduction in the complexity of the assembly problem. The task of resolving such complex repeats is left to specialized modules that use additional sources of information.

It is important to discuss at this moment the task of identifying those reads that belong to repetitive regions. The techniques of Kececioglu and Yu and Tammi et al. are greatly helped by such information. The most common methods for repeat identification use the very nature of the shotgun sequencing project. The reader is reminded that a shotgun sequencing project starts through the random shearing of the DNA into a collection of fragments whose ends are then sequenced such that they over-sample the initial DNA to a specified extent (ranging from 5 to 10 times for typical sequencing projects). For a bacterial genome each base is thus expected to appear in 8 reads (corresponding to $8\times$ coverage). The DNA of a repeat is over-sampled in proportion to the number of copies. One can, therefore, expect that each base of a two copy repeat would occur in approximately 16 reads. This simple idea is very effective not only in identifying repeats but also in estimating their specific copy number [104]. From a theoretical standpoint, the random nature of the initial shearing process allows the development of statistical tests to identify the repetitive sequences. Kececioglu and Yu [99] identify collapsed repeats by estimating the probability of observing a certain depth of coverage at a particular point in the genome. Myers et al. [8] analyze the **arrival rate** of fragments, i.e., the distribution of the fragment start points. Thus they compute the log-ratio of the probability that the observed distribution is representative of a unique, versus the probability that it is representative of a two-copy collapsed repeat. Please note that such statistical approaches rely on a random distribution of the fragments being generated by the shotgun process. Achieving such randomness is difficult in practice leading to limitations in the ability to correctly identify repeats. As an example, during the finishing stages complex regions of a genome are sequenced to a higher coverage than the rest, causing the assembler to incorrectly label them as repetitive.

The methods described so far are not fool-proof, leading to mis-assembled repeats or to the failure to assemble the reads corresponding to large copy-number repeats (due to the k-mer frequency thresholding method described above). Statistical approaches are generally poorly suited to identifying the difference between repeats with low copy numbers and are confused by skewed fragment distributions due to an imperfect shotgun process. The error correction techniques that rely on a multiple alignment of reads to identify errors require a certain amount of coverage in order to correctly distinguish the sequencing errors (generally, 4 reads are required to confidently identify a distinguishing column). Unfortunately certain classes of repetitive sequences are under-represented in the shotgun libraries [37] thus escaping detection. It is, therefore, important for assemblers to rely on additional sources of information when identifying and correctly assembling repeats. The Euler assembler [60] iden-

tifies the effect of repeats on the structure of the SBH graph, a situation they call a **tangle** (Fig. 14). Identifying such regions that cannot be unambiguously resolved by the assembler allows the design of specific laboratory experiments meant to provide the additional information needed for "untangling" the graph [46]. Other assemblers make use of the "mate-pair" information by linking together reads from opposite ends of the same fragment. The presence of conflicts in the mate-pair data is usually a good indication for the existence of a repeat [18], even in the cases when statistical tests are inconclusive. Mate-pairs are thus used to guide the assembly process [5,8,105] or to identify and repair incorrectly assembled contigs [19,20,64]. Arachne [19] identifies "weak" regions of the contigs, i.e., regions supported only by fragment overlaps and not mate-pairs, then breaks such contigs in order to avoid the potential mis-assembly.

Figure 21 highlights three common scenarios for mis-assemblies caused by repeats (represented in different shades of gray in the figure). For each mis-assembly scenario we indicate in gray those mate-pair relationships that become invalidated. These can be used as an indicator of mis-assembly. In the case of collapsed tandem repeats (Fig. 21(a)) mate pairs linking distinct repeat copies become too short, or force the reads to be incorrectly oriented with respect to each other. The situation when a collapsed repeat forces the excision of a contig (Fig. 21(b)) leads to mate-pair links connecting the middle of a contig with another one. Such situations are handled by the "positive breaking" routine of Arachne. Finally, the rearrangement of the genome around repeats (Fig. 21(c)) may lead to a lengthening of the mate-pairs (as shown by the mate-pair a in the figure). This last example also shows one of the possible pitfalls of using mate-pair data to guide the assembly process. The genome can be mis-assembled in such a way as to preserve all the mate-pair relationships (as shown by the links drawn in black). An assembler that uses mate-pairs to guide the placement of reads may thus inadvertently re-arrange the genome without providing any evidence of mis-assembly. Note that the last example is not a purely theoretical one. Such a situation occurs in the assembly of bacteria where ribosomal RNA genes (3–5 kbp repeats) commonly lead to such rearrangements.

4.4 Consensus Generation

The desired output of a shotgun sequence assembler is a tiling of the reads corresponding to their correct location along the genome, together with an estimate of the base composition of the original DNA strand. In the ideal case when the reads contain no errors, the original DNA sequence is easily inferred from the tiling of reads. The situation is more complicated when sequencing errors or mis-assembled reads (errors in the tiling) are present. In this case the problem of identifying the

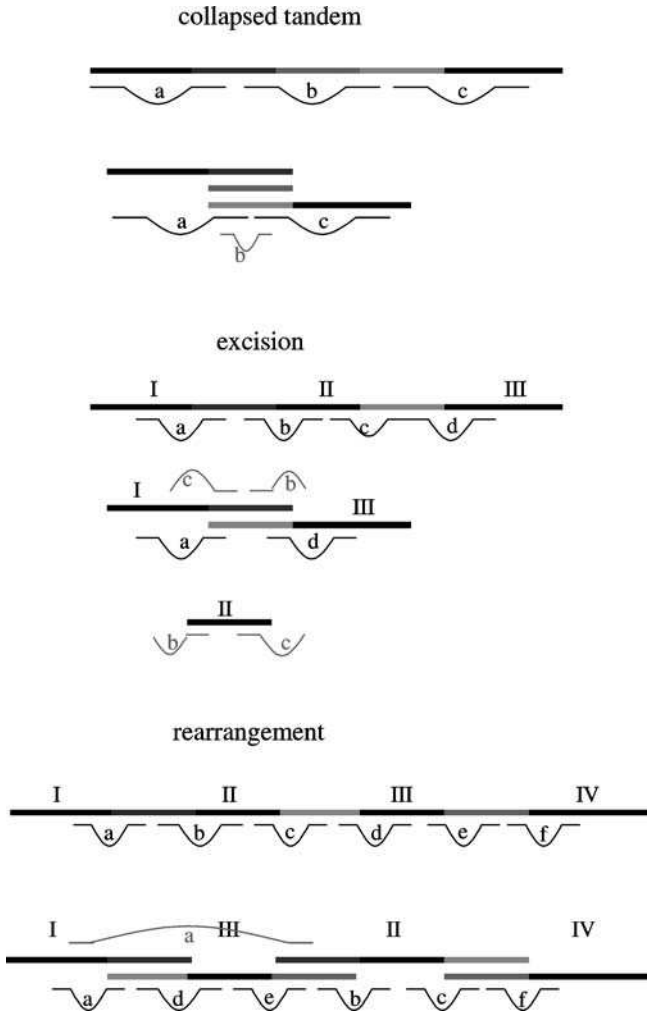


FIG. 21. Types of misassembled repeats a) collapsed tandem; b) excision; c) genome rearrangement.

sequence of the DNA molecule corresponds to the well studied problem of multiple sequence alignment (see, for example, [87]). For a set of sequences, the goal is to identify a "best" multiple alignment under a specified definition of alignment quality. In the case of sequence assembly the objective function for the multiple alignment involves the **consensus sequence**, a representation of the DNA being sequenced. This

problem requires finding a consensus sequence *S* such that the sum of the distances (in terms of standard edit-distance measures) between all reads and *S* is minimized. In the general case there are no known efficient methods to compute the consensus sequence. In the case of shotgun sequence assembly, however, the low number of errors in the sequencing reads implies that most heuristics lead to good solutions.

Most assembly algorithms follow an iterative approach to computing the consensus sequence. Such algorithms start with one of the reads as the consensus, then iteratively refine this consensus by adding the other reads, one by one, to the already computed alignment. The multiple alignment problem is, in this fashion, reduced to the computation of several pair wise alignments, a much simpler task. In the case of algorithms following the greedy paradigm, the order of the addition of fragments is naturally defined by the order in which the algorithm examines fragment overlaps. In fact, in most greedy algorithms [3,5,64,79,90,92] the layout and consensus stages of the assembler are combined by maintaining a correct consensus for each intermediate contig. This approach allows the pair wise alignment routine to take into account the sequence of the already computed consensus. In addition to the consensus, TIGR Assembler [5] also keeps track of the characteristics of the multiple alignment by storing for each location in the consensus the **profile** [106] of all distinct bases that align to that consensus location. The profile consists of a list of all distinct bases occurring at that particular column, together with their multiplicities.

Algorithms following the **overlap-layout-consensus** paradigm start the consensus stage with a rough estimate of the location of each read in the final multiple alignment. The consensus algorithm can use this information to guide the iterative alignment procedure [8,18,58,91]. Phrap [91] uses the alignments between reads defined by the greedy stage to define a graph connecting the perfectly matching portions of these alignments. The final consensus represents a mosaic of the reads obtained from a maximum weight path in this graph. The algorithm used in AMASS [90] relies on the low error rates in the fragments to identify columns of the multiple alignments where all reads agree, then performs a multiple-alignment routine only in those sections located between exact matches, leading to a very efficient algorithm. Kececioglu and Myers [58] define the consensus computation as the **maximum weight trace problem**, by constructing a graph whose edges correspond to the bases matched by the pair wise overlap stage of the algorithm, and requiring that the order of the bases, as defined by each individual read, is preserved in the consensus. This problem is NP-complete, thus they propose a sliding window heuristic for computing the consensus.

These heuristic approaches often produce imperfect alignments, since the quality of the alignment is affected by the order in which the fragments are added to the alignment. Anson and Myers [107] propose an iterative algorithm for optimizing the rough alignments produced by the consensus algorithm. Their approach removes

one read r at the time from alignment A then realigns r to $A - \{r\}$ in an attempt to decrease the number of errors. A similar technique was also used by Huang [92] as part of the CAP assembler.

As a final step, most consensus algorithms attempt to determine the quality of the consensus sequence. Such quality estimates are commonly produced for each read by the base-calling software [96] and Churchill and Waterman [108] propose a statistical model that combines these error probabilities into an estimate of consensus quality in each column of the multiple alignment. Their algorithm also provides a method for deciding on the specific base-call for each consensus base. Previously [2], the consensus base was computed through a simple majority rule. Bonfield and Staden [109] suggest a parametric approach to consensus computation that can be tuned to take into account situations encountered in real data.

All these techniques for assessing the quality of the consensus sequence generated by the assembler make the assumption that each column in the multiple alignment corresponds to a unique base in the sequenced DNA strand. In other words, all the bases in a column must be identical with the exception of differences caused by sequencing errors. Columns that contain a mixture of bases are considered evidence of mis-assembly and are therefore assigned a low quality. Such low-quality bases are usually targeted as part of the finishing process [44,45]. In Section 5.1 we will discuss a situation in which this basic assumption is not true, specifically the case when the DNA being sequenced consists of a mixture of two or more highly similar molecules.

4.5 Scaffolding

With the exception of very simple data-sets, assembly programs are unable to correctly reconstruct the genome as a single contig. The output of an assembler usually consists of a collection of contigs whose placement along the genome is unknown. There are three main reasons that lead to the inability of an assembler to join together all the reads into a single contig:

- The random sampling of DNA fragments from the genome naturally leads to certain sections not sampled by any sequencing reads. Even at the levels of coverage selected for typical sequencing projects (5–10×), the probability of observing such gaps is relatively high, especially when considering that short overlaps between reads (usually less than about 40 bp) cannot be reliably detected by the assembly software.
- Certain regions of the genome are poorly represented in the fragment libraries due to purely biological reasons.
- The assembler may not be able to correctly assemble repeats leading to portions of the genome that either remain un-assembled, or are assembled incorrectly.

Since the ultimate goal of assembly is to reconstruct, as much as possible, the original structure of the genome, scientists have developed techniques meant to identify the correct placement of these genomes along the genome. One such technique, called **scaffolding** [34] orders and orients the contigs with respect to each other using the information contained in the pairing of reads sequenced from opposite ends of a fragment (see Fig. 5). This technique was used for the first time to guide the assembly and finishing of *Haemophilus influenzae* [4], leading to the first complete sequence of a free-living organism.

In abstract terms, the mate-pair relationship between reads implies a linking of two contigs with respect to their relative order and orientation. Scaffolding can thus be extended to take into account other sources of information defining a particular relative placement of the contigs. Some such sources of information are:

- **contig overlaps**—ideally the output of an assembler should consist in a collection of non-overlapping contigs. Sequencing errors, often situated at the end of reads, lead to contigs that can not be merged by the assembler. These overlaps can be identified by less stringent alignment algorithms and provide valuable scaffolding information.
- **physical maps**—for many genomes scientists map the locations of known markers along the genome. This is, for example, an essential step in a hierarchical BAC-by-BAC sequencing project. The location of these markers in the assembled contigs provides information useful in anchoring the contigs to known positions along the genome [110].
- **alignments to a related genome**—an increasing number of finished genomes is becoming available to the scientific community. Thus for many organisms it is possible to obtain the complete sequence of a closely related organism. The alignment of the contigs to this reference can thus be used in those cases when physical maps are not available. This information should, however, be used with care since genomic rearrangements may lead to an incorrect reconstruction of the genome.
- **gene synteny data**—in many organisms certain genes occur in clusters. This information can be used for scaffolding by identifying, for example, pairs of contigs that contain genes belonging to a same cluster. While the orientation of the contigs cannot generally be determined, their spatial proximity is information useful in scaffolding.

Please note that some sources of linking data are inherently erroneous and may only provide an approximate estimate of contig adjacency. As an example, physical maps provide only a coarse level of detail as the distances between markers can only be approximately determined. Similarly, gene synteny data provides little information about the distance between contigs.

This variety of sources of information can be used to infer the relative placement of two contigs, yielding a set of abstract links between adjacent contigs. Each link defines one or more of the following constraints on the relative placement of the two contigs:

- **ordering**—one of the contigs occurs "before" the other in the sense of a linear or circular order corresponding to the location along a linear or circular chromosome;
- **orientation**—the specification of whether the two contigs represent samples from the same or from opposite strands of the double-stranded DNA;
- **spacing**—an indication of the distance between the two contigs.

The scaffolding problem can, therefore, be defined as:

> Given a set of contigs and a set of pair wise constraints, identify a linear (circular in the case of most bacterial genomes) embedding (defining both the order and the orientation of the contigs along a chromosome) of these contigs such that most constraints are satisfied.

In the general case this problem is intractable [66,111]. Interestingly, even when relaxing the problem by ignoring the ordering of the contigs, the associated contig orientation problem is also intractable [58], as is the complementary problem of ordering the contigs when a proper orientation is given. The orientation problem is equivalent to finding a maximum bipartite sub-graph, while the ordering problem is similar to the Optimal Linear Arrangement problem, both of which are NP-hard [50]. Kececioglu and Myers [58] describe a greedy approximation algorithm to the orientation problem in the context of sequence assembly that achieves a solution at most twice worse than the optimal.

All constraints defined above can be described in terms of linear inequalities and, therefore, the scaffolding problem can be formulated as a constraint satisfiability problem [19,112]. Due to the complexity of solving such problems (typical solutions involve many iterations of complex relaxation steps) practical implementations of this approach are limited to local optimization steps within the scaffolder [8,19,66]. As an example, the scaffolder used by Celera Assembler [8] refines the placement of the contigs by attempting to minimize the "stretch" of the mate-pair relationships as defined by the sum of the squares of deviations from the mean fragment size. In many cases this restricted problem can be easily solved as it reduces to a system of linear equations.

Most practical solutions to the scaffolding problem use a graph-theoretical approach. With one exception, the Eulerian graph approach of Pevzner et al. [61], all scaffolding algorithms to date construct a graph whose nodes correspond to contigs

and whose edges correspond to the presence of contig links between the corresponding contigs. In order to reduce the effect of errors scaffolders require at least two links between adjacent contigs. They then "bundle" all links between adjacent contigs into a single contig edge and greedily join the contigs into scaffolds. The path-merging algorithm of Huson et al. [111] examines the edges in decreasing order of the number of links in the bundle. Whenever an edge links two distinct scaffolds, the algorithm attempts to merge the scaffolds together (hence the name: path-merging). Arachne [18,19] uses edge weights that depend on both the number of links and the size of the edge, and Phusion [20] examines edges in order of their lengths, from smallest to largest. The Bambus scaffolder [113] allows the user to specify the order in which links are considered in terms of both library size and edge weight. Arachne [19] and Jazz [22] incorporate an iterative error-correction step during which scaffolds may be broken then re-combined based on links that were not used during the original greedy step. Note that Bambus is the only scaffolder that can currently use all the types of linking information described at the beginning of this section. All other scaffolders use only the mate-pair information.

4.6 Assembly Validation

One of the most important assembly tasks is that of validating the result produced by the assembler. Well defined methods for assembly validation are important as a debugging tool during the development of new assembly software [8,18,60,64,79, 90,92]. Such methods are an invaluable resource in comparing the performance of different assembly algorithms [12,114–117] and are essential to machine-learning approaches to shotgun-sequence assembly. It is, therefore, surprising that, to our knowledge, no comprehensive benchmark exists that allow an objective comparison of assembly programs.

In most cases the validation is performed through the use of artificially generated data-sets. Such data allow the developers to control the characteristics of the input to the assembler and also provide a simple way to assess the accuracy of the assembly output. The programs used to simulate a shotgun sequencing project [118,119] allow the users to specify the sequencing error rate, read and fragment sizes, and even allow the generation of complex repeat structures [119]. These assembly "simulators" can provide information regarding the placement of the fragments, thus allowing one to test not only if the consensus sequence produced by the assembler matches that of the test data-set, but also if the reads are assembled in the correct place. The use of such artificial data allows a complete verification of the correctness of an assembly algorithm. The ultimate test, however, requires the use of real data as they contain features difficult or impossible to emulate in a controlled setting.

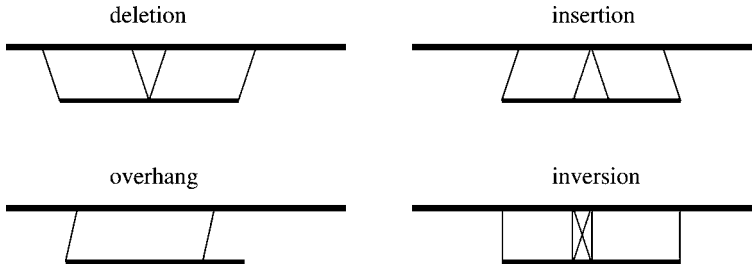


FIG. 22. Possible mis-assemblies identified by alignment to the genome.

When using real data it is difficult to know the correct reconstruction of the genome, in particular it is generally impossible to know the correct placement of all reads along the genome. The validation of assemblies based on real shotgun sequence data requires one or more of the following types of information:

- an independently verified consensus sequence,
- knowledge of the location of experimentally identified markers,
- the assembly output from a different assembly program,
- high-quality mate-pairing data.

The first two classes of information are useful in the testing of assembly algorithms during development and in comparing different assembly algorithms. The latter two classes of information can be used to validate the results of *ab-initio* sequencing projects, thus are most useful in the day-to-day use of assembly programs.

When a correct consensus sequence is available, whether artificially generated or from a finished genome, assessing the quality of an assembly is relatively straightforward. Assembly errors manifest themselves as contigs that do not align perfectly to the consensus (for some examples see Fig. 22). If the reference represents the sequence of a finished genome it is important to note that the chance exists that the finished sequence may actually be incorrect.

Certain genetic markers, such as **sequence tag sites (STS)**[12] [67] can be used to validate the global structure of the assembly when no finished reference is available. The sequences of the markers are known, thus their locations within the assembly can be easily identified. This *in silico* map of the markers is then compared to the map obtained through laboratory experiments in order to validate the overall structure of the assembly [110]. Such an approach was used by both Celera [9] and the Human

[12]**sequence tag sites (STS)**—short (200–500 bp) sequences of DNA that occur at a single place within the genome and whose location in the genome can be mapped experimentally. They serve as landmarks along the genome.

Genome Consortium [120] to ascertain the quality of their assemblies of the human genome. This method provides information only about the large-scale structure of the assembly and is not able to identify small mis-assemblies. Furthermore, physical maps often contain errors and should not entirely be relied upon.

The comparison of different assemblies (either different algorithms/programs or different algorithm parameters) is difficult to interpret. Without additional information it is generally difficult to identify which assembly is incorrect, though the comparison is an important first-step in identifying regions of the genome that require further investigation. It is important to note that overall assembly statistics (such as average and maximum contig and scaffold sizes) are not appropriate measures of assembly quality. Large contig sizes can easily be achieved at the expense of mis-assemblies. One should therefore ignore such statistics if not accompanied by an assessment of the correctness of the assembly.

The validation of assemblies is possible even without independent certificates, such as mapping data or completed genome sequence. Several characteristics of the shotgun-sequencing process can be used to detect possible mis-assemblies. In Section 4.2 we described how deep fragment coverage can be an indicator of the collapse of repeats. Conversely, low coverage regions may indicate a mis-assembly due to a short repeat. These approaches identify deviations from the expected distribution of fragments produced by a purely random shotgun sequencing process, and are, therefore, sensitive to the problems caused by biases in the initial distribution. A more reliable source of information are the mate-pair relationships between reads. This information was also recently proposed as a sensitive method of identifying structural rearrangements in the human genome that are related to various cancers [121]. Note, however, that many assembly algorithms use the mate-pair information to guide the assembly process, thus rendering such information of limited use for validation.

An excellent summary of all these methods of assembly validation is presented by Huson et al. [117] in the context of comparing two assemblies of the same set of reads. They identify not only tell-tale signs of misassemblies, but also propose expressive visualization schemes that allow the inspection of large assemblies.

5. Exotic Assembly

Up to this point we have presented solutions to the most common problems related to shotgun sequence assembly. These algorithms contributed to the current genomic revolution leading to an exponentially increasing number of genomes being sequenced. This increase in the numbers and types of genomes that are analyzed is uncovering new problems to be solved by assembly programs. In this section we will briefly discuss a few of the current assembly challenges.

5.1 Polymorphism Identification and Haplotype Separation

All the algorithms described so far assume that the input to the sequencing process consists of a single DNA molecule, and the only complexities are introduced by repeats and sequencing errors. It is often the case in practice that the source of DNA consist of two or more highly similar molecules. Such is the situation in most eukaryotic organisms where each chromosome usually occurs in two copies, one inherited from each parent. The two copies of each chromosome are often similar enough that the assembler is not able to separate their corresponding reads. In effect we are encountering the case of a large over-collapsed repeat that spans an entire chromosome. Much like in the case of repeats the two copies of a chromosome contain a number of **polymorphisms**, sites where the copies differ. Commonly observed differences consist of: **single nucleotide polymorphisms** (**SNPs**)—single nucleotide differences between the chromosomes; **indels**—insertions or deletions of more than 2 base pairs; **inversions**—sections of DNA that occur in different orientation in the two chromosomes; and **translocations**—sections of DNA that occur in different orders along the chromosome. These differences, in addition to their effects on assembly, are important to identify since they often represent the genetic elements of many diseases such as cancers or genetic disorders.

The large scale polymorphisms (large indels, inversions, and translocations) can be detected by analyzing inconsistencies in mate-pair data [121]. SNPs and short indels are more easily identified from the multiple alignments produced during the consensus stage of assembly. Potential SNPs correspond to columns of the multiple alignment that contain two or more high-quality bases that disagree (see Fig. 23). By quality we refer to the base-calling error estimates provided by sequencing software. This approach, suggested by Taillon-Miller et al. [122] was used to identify SNPs in the recently sequenced human genome [123–125]. In order to reduce the number of false-positives caused by sequencing errors Altshuller et al. [123] propose to restrict their analysis to only those bases surrounded by high-quality sequence. Their **neighborhood quality standard (NQS)** requires that both the base defining a SNP

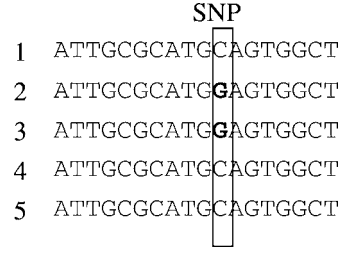


FIG. 23. Column in a multiple-alignment indicating a potential SNP site.

and the five bases surrounding it on either side be of high quality. They show that this simple approach greatly reduced the number of false positives. Similarly, the use of base qualities is suggested by Read et al. [126] in the context of identifying SNPs between two highly similar strains of *Bacillus anthracis*, the bacterium causing anthrax. Their technique provides an estimate of the probability of each SNP being correct by computing the consensus quality for each of the two variants, then choosing the lower value as the quality of the SNP.

It is often important not only to identify the polymorphic sites, but also determine which sites belong to the same chromosome, a process called **haplotyping**. In Section 4.2 we discussed how correlated differences between reads can provide enough information to separate out different copies of a repeat. The same techniques can be used to separate the two distinct chromosomes, though, in general, the data can only be partially separated as long regions of similarity between the haplotypes break the connection between consecutive SNPs. Lancia et al. [127] define this problem in terms of **fragment conflict graphs**. This graph represents each read as a node, and connects two nodes if the corresponding reads disagree at one or more SNP sites. In the absence of sequencing errors, the fragment conflict graph is bipartite, corresponding to the two haplotypes. In the presence of sequencing errors the graph can be transformed into a bipartite graph (thereby separating the haplotypes) by either removing a set of reads, or removing a set of SNPs (effectively marking them as sequencing errors). Thus they define three optimization problems:

Minimum fragment removal—remove the minimum number of fragments such that the remaining conflict graph is bipartite

Minimum SNP removal—remove the minimum number of SNP sites such that the remaining graph is bipartite

Longest haplotype reconstruction—remove a set of fragments such that the sum of the lengths of the derived haplotypes is maximized.

They proceed to show that all these problems are NP-hard in the general case, however they can be efficiently solved in the case when the reads do not contain any gaps, the situation often encountered in practice. Lippert et al. [128] further extend the results of Lancia et al. by examining the complexities introduced by the possibility of multiple optimal solutions to the problems described above. Thus they suggest the need for a better formulation of the separation problem that does not depend on the choice of an optimal solution from among a number of alternatives.

It is important to discuss the effect of haplotypes on the assembly process itself. Sequencing errors may lead to inconsistencies in the structure of the read overlaps. Such inconsistencies are only accentuated by the presence of distinct haplotypes in the shotgun data, leading to characteristic **bubbles** (see Fig. 24) in the overlap graph. Fasulo et al. [129] describe the algorithms used by Celera Assembler that allow the

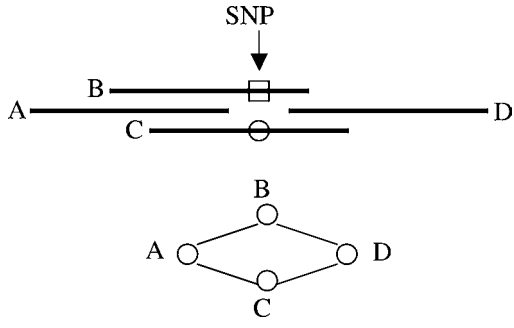


FIG. 24. Characteristic "bubble" in the overlap graph (bottom) caused by a SNP. Reads B and C differ at the SNP location leading to the absence of an overlap edge between them.

assembly of divergent haplotypes into a single consensus sequence. Note that rather than attempting to separate the different haplotypes they collapse them into a single multiple alignment, leaving the task of haplotype separation to specialized tools such as those described above.

Currently there are no widely accepted ways of analyzing and representing haplotypes other than SNPs. This area of research is of great importance as more organisms are being assembled and we encounter the need to understand complex polymorphism events.

5.2 Comparative Assembly

The number of completed genomes is constantly increasing, especially in the case of bacterial genomes. This information can be used to guide the assembly of related genomes in a process called **comparative assembly**. In Section 4.5 we already discussed the use of such information in guiding scaffolding. For many bacterial genomes such an approach allows the construction of scaffolds that span entire chromosomes [113], thus greatly reducing the amount of work necessary to finish these genomes.

In addition, it is often the case that multiple closely related strains of the same bacterium are sequenced in order to identify polymorphic sites [126]. Read et al. used sequencing in a forensic setting to identify differences between bacteria formerly classified within the same strain, however such studies are also essential in identifying markers for pathogen identification or for short-term evolutionary analyses. The availability of the complete sequence of a closely related organism also allows a more efficient design of the shotgun sequencing process. A typical shotgun sequencing project requires an 8 to 10 over sampling of the DNA in order to minimize the

number of gaps in the resulting sequence, and to allow the creation of large scaffolds. However, when a reference genome sequence is available it is conceivable to perform a considerably lower amount of sequencing (3–5 times over-sampling). The goal in this case is to generate contigs large enough to allow an unambiguous mapping to the reference sequence. Not only is the scaffolding problem not an issue in this case, but also the remaining gaps in the sequence can be quickly closed through direct sequencing experiments since their location is uniquely identified by the mapping to the reference. The costs of sequencing a genome can, thus, be greatly reduced, making feasible the survey sequencing of multiple individuals from the same strain.

To our knowledge, no assembly algorithm fully exploits the comparative information, though a few make limited use of this information.

5.3 Multiple Organisms

Section 5.1 discussed the issues involved when sequencing eukaryotic genomes where each chromosome occurs in two distinct copies. Such assembly problems can be naturally extended to the problem of assembling data from multiple similar, but not identical, organisms. For example, when sequencing non-clonal bacterial populations, or when the organism being sequenced contains multiple divergent copies of a same plasmid [130]. Furthermore, sequencing has been proposed as a method for assessing the diversity of bacterial populations in various environments [131,132], though, until recently, such surveys performed targeted sequencing of known markers, such as the **16S ribosomal RNA**[13] subunits. Reductions in sequencing costs allow wider scale surveys by shotgun-sequencing an entire bacterial population. Such efforts are currently under way at TIGR (http://www.tigr.org) to sequence the bacteria in the human gastro-intestinal tract [133] and at IBEA (http://www.bioenergyalts.org) to sequence the bacteria found in the Sargasso sea [134].

Current assemblers are not well suited to handle the assembly of such mixed populations. The goal of the assembler is, in this case, a collection of contigs corresponding to each of the individual organisms being assembled. This is not a new problem, as the shotgun method was applied to the sequencing of whole genomes that contain multiple chromosomes. While the multiple chromosomes of an organism are generally equally covered by the set of fragments, bacterial populations generally consist of an un-even mixture of organisms. Thus a few of the bacteria end up being heavily sampled, while others are represented in the shotgun library at a very low coverage. This fact renders useless any statistical approaches to repeat identification as the

[13] **16S ribosomal RNA subunit**—gene encoding a section of the ribosome. This subunit can be selectively amplified in most bacteria thus providing a small sequence (1.5 kb) that can be used in comparative analyses.

relative distribution of bacteria can not generally be modeled. The uneven level of coverage also limits the use of standard procedures for repeat and haplotype separation since these techniques generally require sufficient sequence coverage from all organisms in order to be effective. New algorithms are therefore necessary to handle the correct assembly of mixed populations. Comparative assembly techniques will also be extremely valuable in making use of the sequence data obtained from the poorly represented members of the population.

5.4 Heterogeneous Assembly

Most of the currently available assemblers are optimized to assemble shotgun data alone, making certain assumptions on the characteristics of these data. These assemblers usually assume that the reads are generally short (despite dramatic improvements in sequencing technologies read lengths have not increased beyond 2000 base pairs). The alignment algorithms can thus afford to pre-allocate the required memory in order to speed up the execution. Furthermore, many assemblers make certain assumptions on the quality of the data, or the availability of quality estimates for the reads. K-mer hashing techniques used to compute overlaps rely on the high similarity between reads, otherwise matching k-mers could not be found.

Assembly algorithms will need to be designed to take advantage of improvements to sequencing technologies and the multiple sources of sequence information currently available in the public databases. New sequencing technologies promise the ability to obtain the sequence of BACs and even chromosomes in a single step, though it is expected that these data will be characterized by large error rates. The availability of such long erroneous reads leads to specific assembly challenges [135]. Furthermore, the data available to the assembler consist not only of shotgun reads, but also finished BAC sequences, expressed sequence tag (EST) sequences [136] or contigs produced by draft-sequencing projects. Scientists at University of California in Santa Cruz had to develop a specialized assembler to use such heterogeneous data in the assembly of the first draft of the human genome [137].

6. Conclusions

The assembly problem was repeatedly considered solved, first when efficient approximation algorithms for the shortest superstring problem became available, again when assembly software was able to routinely assemble entire bacterial genomes, and recently when software exists that can assemble entire mammalian genomes in a relatively short time. Continued reductions in sequencing costs have led to a dramatic increase in the numbers of genomes being sequenced. A direct effect of this

genomic revolution is the uncovering of novel uses for assembly programs, leading to new algorithmic challenges such as those discussed in Section 5. We therefore hope that this survey will provide an adequate starting point for those interested in further exploring the algorithmic and practical problems arising in this dynamic field.

ACKNOWLEDGEMENTS

I would like to thank Art Delcher, Adam Phillippy, and Steven Salzberg for their useful comments and continued support. This work was supported in part by the National Institutes of Health under grant R01-LM06845.

REFERENCES

[1] Sanger F., et al., "Nucleotide sequence of bacteriophage lambda DNA", *J. Mol. Biol.* **162** (4) (1982) 729–773.
[2] Staden R., "Automation of the computer handling of gel reading data produced by the shotgun method of DNA sequencing", *Nucleic Acids Res.* **10** (1982) 4731–4751.
[3] Gingeras T.R., et al., "Computer programs for the assembly of DNA sequences", *Nucleic Acids Res.* **7** (2) (1979) 529–545.
[4] Fleischmann R.D., et al., "Whole-genome random sequencing and assembly of Haemophilus influenzae Rd", *Science* **269** (5223) (1995) 496–512.
[5] Sutton G.G., et al., "TIGR assembler: A new tool for assembling large shotgun sequencing projects", *Genome Science and Technology* **1** (1995) 9–19.
[6] Green P., "Against a whole-genome shotgun", *Genome Res.* **7** (5) (1997) 410–417.
[7] Adams M.D., et al., "The genome sequence of Drosophila melanogaster", *Science* **287** (5461) (2000) 2185–2195.
[8] Myers E.W., et al., "A whole-genome assembly of Drosophila", *Science* **287** (5461) (2000) 2196–2204.
[9] Venter J.C., et al., "The sequence of the human genome", *Science* **291** (5507) (2001) 1304–1351.
[10] Consortium I.H.G.S., "Initial sequencing and analysis of the human genome", *Nature* **409** (2001) 860–921.
[11] Semple C.A., et al., "Computational comparison of human genomic sequence assemblies for a region of chromosome 4", *Genome Res.* **12** (3) (2002) 424–429.
[12] Aach J., et al., "Computational comparison of two draft sequences of the human genome", *Nature* **409** (6822) (2001) 856–859.
[13] Adams M.D., et al., "The independence of our genome assemblies", *Proc. Natl. Acad. Sci. USA* **100** (6) (2003) 3025–3026.
[14] Waterston R.H., Lander E.S., Sulston J.E., "More on the sequencing of the human genome", *Proc. Natl. Acad. Sci. USA* **100** (6) (2003) 3022–3024, author reply 3025–3026.

[15] Waterston R.H., Lander E.S., Sulston J.E., "On the sequencing of the human genome", *Proc. Natl. Acad. Sci. USA* **99** (6) (2002) 3712–3716.
[16] Green P., "Whole-genome disassembly", *Proc. Natl. Acad. Sci. USA* **99** (7) (2002) 4143–4144.
[17] Myers E.W., et al., "On the sequencing and assembly of the human genome", *Proc. Natl. Acad. Sci. USA* **99** (7) (2002) 4145–4146.
[18] Batzoglou S., et al., "ARACHNE: a whole-genome shotgun assembler", *Genome Res.* **12** (1) (2002) 177–189.
[19] Jaffe D.B., et al., "Whole-genome sequence assembly for Mammalian genomes: arachne 2", *Genome Res.* **13** (1) (2003) 91–96.
[20] Mullikin J.C., Ning Z., "The phusion assembler", *Genome Res.* **13** (1) (2003) 81–90.
[21] Havlak P., et al., "The Atlas whole-genome assembler", in: *Currents in Computational Molecular Biology, Montreal, Canada*, 2001.
[22] Aparicio S., et al., "Whole-genome shotgun assembly and analysis of the genome of Fugu rubripes", *Science* **297** (5585) (2002) 1301–1310.
[23] Waterston R.H., et al., "Initial sequencing and comparative analysis of the mouse genome", *Nature* **420** (6915) (2002) 520–562.
[24] Consortium R.g.s., "Rat. Genome project", http://www.hgsc.bcm.tmc.edu/projects/rat.
[25] Kirkness E.F., et al., "The dog genome: survey sequencing and comparative analysis", *Science* **301** (5641) (2003) 1898–1903.
[26] Dehal P., et al., "The draft genome of Ciona intestinalis: insights into chordate and vertebrate origins", *Science* **298** (5601) (2002) 2157–2167.
[27] Green E.D., "Strategies for the systematic sequencing of complex genomes", *Nat. Rev. Genet.* **2** (8) (2001) 573–583.
[28] Cai W.W., et al., "A clone-array pooled shotgun strategy for sequencing large genomes", *Genome Res.* **11** (10) (2001) 1619–1623.
[29] Lander E.S., Waterman M.S., "Genomic mapping by fingerprinting random clones: A mathematical analysis", *Genomics* **2** (3) (1988) 231–239.
[30] Czabarka E., et al., "Algorithms for optimizing production DNA sequencing", in: *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000.
[31] Batzoglou S., et al., "Sequencing a genome by walking with clone-end sequences: A mathematical analysis", *Genome Res.* **9** (1999) 1163–1174.
[32] Li X., Waterman M.S., "Estimating the repeat structure and length of DNA sequences using ell-tuples", *Genome Res.* **13** (8) (2003) 1916–1922.
[33] Arratia R., et al., "Genomic mapping by anchoring random clones: A mathematical analysis", *Genomics* **11** (4) (1991) 806–827.
[34] Roach J.C., et al., "Pairwise end sequencing: A unified approach to genomic mapping and sequencing", *Genomics* **26** (1995) 345–353.
[35] Port E., et al., "Genomic mapping by end-characterized random clones: A mathematical analysis", *Genomics* **26** (1) (1995) 84–100.
[36] Yeh R.F., et al., *Predicting Progress in Shotgun Sequencing with Paired Ends*, 2002.
[37] Chissoe S.L., et al., "Representation of cloned genomic sequences in two sequencing vectors: correlation of DNA sequence and subclone distribution", *Nucleic Acids Res.* **25** (15) (1997) 2960–2966.

[38] Mullis K., et al., "Specific enzymatic amplification of DNA in vitro: the polymerase chain reaction", *Cold Spring Harb. Symp. Quant. Biol.* **51 Pt 1** (1986) 263–273.
[39] Burgart L.J., et al., "Multiplex polymerase chain reaction", *Mod. Pathol.* **5** (3) (1992) 320–323.
[40] Tettelin H., et al., "Optimized multiplex PCR: efficiently closing a whole-genome shotgun sequencing project", *Genomics* **62** (3) (1999) 500–507.
[41] Beigel R., et al., "An optimal procedure for gap closing in whole genome shotgun sequencing", in: *Proceedings of the Fifth Annual International Conference on Computational Biology (RECOMB)*, 2001.
[42] Alon N., et al., "Learning a hidden matching", in: *Proceedings of the IEEE symposium on Foundations of Computer Science (FOCS)*, 2002.
[43] Staden R., Judge D.P., Bonfield J.K., "Sequence assembly and finishing methods", *Methods Biochem. Anal.* **43** (2001) 303–322.
[44] Gordon D., Desmarais C., Green P., "Automated finishing with autofinish", *Genome Res.* **11** (4) (2001) 614–625.
[45] Gordon D., Abajian C., Green P., "Consed: A graphical tool for sequence finishing", *Genome Res.* **8** (1998) 195–202.
[46] Mulyukov Z., Pevzner P.A., "EULER-PCR: finishing experiments for repeat resolution", *Pac. Symp. Biocomput.* (2002) 199–210.
[47] Toth G., Gaspari Z., Jurka J., "Microsatellites in different eukaryotic genomes: survey and analysis", *Genome Res.* **10** (7) (2000) 967–981.
[48] Myers E.W., "Toward simplifying and accurately formulating fragment assembly", *J. Comp. Bio.* **2** (2) (1995) 275–290.
[49] "HUGO, Summary of the report of the second international strategy meeting on human genome sequencing", http://www.gene.ucl.ac.uk/hugo/bermuda2.html.
[50] Garey M.R., Johnson D.S., *Computers and Intractability*, W.H. Freeman, New York, 1979.
[51] Blum A., et al., "Linear approximation of shortest superstrings", in: *Proc. 23rd Annual Symposium on the Theory of Computing, New Orleans, LA*, 1991.
[52] Peltola H., et al., "Algorithms for some string matching problems arising in molecular genetics", in: *Proc. Information Processing*, 1983.
[53] Teng S., Yao F.F., "Approximating shortest superstrings", *SIAM J. Computing* **26** (2) (1997) 410–417.
[54] Armen C., Stein C., "A 2 2/3-approximation algorithm for the shortest superstring problem", in: *Proc. Combinatorial Pattern Matching*, 1996.
[55] Kececioglu J.D., *Exact and Approximation Algorithms for DNA Sequence Reconstruction*, University of Arizona, 1991.
[56] Smith T.F., Waterman M.S., "Identification of common molecular subsequences", *J. Mol. Biol.* **147** (1) (1981) 195–197.
[57] Peltola H., Soderlund H., Ukkonen E., "SEQAID: a DNA sequence assembling program based on a mathematical model", *Nucleic Acids Res.* **12** (1) (1984) 307–321.
[58] Kececioglu J.D., Myers E.W., "Combinatorial algorithms for DNA sequence assembly", *Algorithmica* **13** (1995) 7–51.

[59] Idury R.M., Waterman M.S., "A new algorithm for DNA sequence assembly", *J. Comp. Bio.* **2** (2) (1995) 291–306.
[60] Pevzner P.A., Tang H., Waterman M.S., "An Eulerian path approach to DNA fragment assembly", *Proc. Natl. Acad. Sci. USA* **98** (17) (2001) 9748–9753.
[61] Pevzner P.A., Tang H., "Fragment assembly with double-barreled data", *Bioinformatics* **17** (Suppl 1) (2001) S225–S233.
[62] Pevzner P.A., Tang H., Waterman M.S., "A new approach to fragment assembly in DNA sequencing", in: *Proceedings of the Fifth Annual International Conference on Computational Biology (RECOMB)*, 2001.
[63] Pop M., Salzberg S.L., Shumway M., "Genome sequence assembly: algorithms and issues", *IEEE Computer* **35** (7) (2002) 47–54.
[64] Huang X., Madan A., "CAP3: A DNA sequence assembly program", *Genome Res.* **9** (1999) 868–877.
[65] Bevan M., et al., "Sequence and analysis of the Arabidopsis genome", *Curr. Opin. Plant. Biol.* **4** (2) (2001) 105–110.
[66] Kent W.J., Haussler D., "Assembly of the working draft of the human genome with GigAssembler", *Genome Res.* **11** (9) (2001) 1541–1548.
[67] Olson M., et al., "A common language for physical mapping of the human genome", *Science* **245** (4925) (1989) 1434–1435.
[68] Wang J., et al., "RePS: a sequence assembler that masks exact repeats identified from the shotgun data", *Genome Res.* **12** (5) (2002) 824–831.
[69] Huson D.H., et al., "Design of a compartmentalized shotgun assembler for the human genome", *Bioinformatics* **17** (Suppl 1) (2001) S132–S139.
[70] Csuros M., Milosavljevic A., "Pooled genomic indexing (PGI) mathematical analysis and experiment design", in: *Proceedings of the 2nd International Workshop on Algorithms in Bioinformatics (WABI)*, Springer-Verlag, 2002.
[71] Parsons R.J., Forrest S., Burks C., "Genetic algorithms, operators, and DNA fragment assembly", *Machine Learning* **21** (1995) 11–33.
[72] Goldberg M.K., Lim D.T., "A learning algorithm for the shortest superstring problem", in: *Proceedings of the Atlantic Symposium on Computational Biology and Genome Information Systems and Technology, Durham, NC*, 2001.
[73] Goldberg M.K., Lim D.T., "Designing and testing a new DNA fragment assembler VEDA-2", http://www.cs.rpi.edu/~goldberg/publications/veda-2.pdf.
[74] Jiang T., Li M., "DNA sequencing and string learning", *Math. Sys. Theory* **29** (1996) 387–405.
[75] King L.M., Cummings M.P., "Satellite DNA repeat sequence variation is low in three species of burying beetles in the genus Nicrophorus (Coleoptera: Silphidae)", *Mol. Biol. Evol.* **14** (11) (1997) 1088–1095.
[76] Kosaraju S.R., Delcher A., "Large-scale assembly of DNA strings and space-efficient construction of suffix trees(Correction)", in: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, STOC'96*, 1996.
[77] Kosaraju S.R., Delcher A.L., "Large-scale assembly of DNA strings and space-efficient construction of suffix trees", in: *Proceedings of the ACM Symposium on the Theory of Computing, STOC'95*, 1995.

[78] Chen T., Skiena S.S., "Trie-based data structures for sequence assembly", in: *Proceedings of the Eighth Symposium on Combinatorial Pattern Matching*, 1997.
[79] Chen T., Skiena S.S., "A case study in genome-level fragment assembly", *Bioinformatics* **16** (2000) 494–500.
[80] Weiner P., "Linear pattern matching algorithms", in: *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, 1973.
[81] Manber U., Myers E.W., "Suffix arrays: A new method for on-line string searches", *SIAM J. Computing* **22** (1993) 935–948.
[82] Ukkonen E., "On-line construction of suffix-trees", *Algorithmica* **14** (1995) 249–260.
[83] McCreight E.M., "A space-economical suffix tree construction algorithm", *J. ACM* **23** (1976) 262–272.
[84] Needleman S.B., Wunsch C.D., "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *J. Mol. Biol.* **48** (1970) 443–453.
[85] Myers E.W., Miller W., "Optimal alignments in linear space", *CABIOS* **4** (1988) 11–17.
[86] Myers E.W., "An O(nd) difference algorithm and its variations", *Algorithmica* **1** (1986) 251–266.
[87] Gusfield D., *Algorithms on Strings, Trees, and Sequences*, The Press Syndicate of the University of Cambridge, 1997.
[88] Altschul S.F., et al., "Basic local alignment search tool", *J. Mol. Biol.* **215** (1990) 403–410.
[89] Pearson W.R., Lipman D.J., "Improved tools for biological sequence comparison", *Proc. Natl. Acad. Sci. USA* **85** (1988) 2444–2448.
[90] Kim S., Segre A.M., "AMASS: A structured pattern matching approach to shotgun sequence assembly", *J. Comp. Bio.* **6** (2) (1999) 163–186.
[91] Green P., *PHRAP documentation: ALGORITHMS*, 1994.
[92] Huang X., "An improved sequence assembly program", *Genomics* **33** (1996) 21–31.
[93] Tammi M.T., et al., "Correcting errors in shotgun sequences", *Nucleic Acids Res.* **31** (15) (2003) 4663–4672.
[94] Roberts M., Hunt B.R., Yorke J.A., Bolanos R., Delcher A., "A preprocessor for shotgun assembly of large genomes", *J. Comp. Bio.*, submitted for publication.
[95] Huang X., et al., "PCAP: A whole-genome assembly program", *Genome Res.* **13** (9) (2003) 2164–2170.
[96] Ewing B., Green P., "Base-calling of automated sequencer traces using phred. II. Error probabilities", *Genome Res.* **8** (3) (1998) 186–194.
[97] Ewing B., et al., "Base-calling of automated sequencer traces using phred. I. Accuracy assessment", *Genome Res.* **8** (3) (1998) 175–185.
[98] Chou H.H., Holmes M.H., "DNA sequence quality trimming and vector removal", *Bioinformatics* **17** (12) (2001) 1093–1104.
[99] Kececioglu J., Yu J., "Separating repeats in DNA sequence assembly", in: *Proceedings of the Fifth Annual International Conference on Computational Biology (RECOMB), Montreal, Canada*, 2001.
[100] Myers E.W., "Optimally separating sequences", *Genome Informatics* **12** (2001) 165–174.

[101] Tammi M.T., et al., "Separation of nearly identical repeats in shotgun assemblies using defined nucleotide positions, DNPs", *Bioinformatics* **18** (3) (2002) 379–388.

[102] Tammi M.T., Arner E., Andersson B., "TRAP: Tandem Repeat Assembly Program produces improved shotgun assemblies of repetitive sequences", *Comput. Methods Programs Biomed.* **70** (1) (2003) 47–59.

[103] Pe'er I., Shamir R., "Spectrum alignment: Efficient resequencing by hybridization", in: *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2000.

[104] Bailey J.A., et al., "Segmental duplications: organization and impact within the current human genome project assembly", *Genome Res.* **11** (2001) 1005–1017.

[105] Pevzner P., Tang H., "Fragment assembly with double barreled data", in: *ISMB'01*, 2001.

[106] Gribskov M., McLachlan A.D., Eisenberg D., "Profile analysis: Detection of distantly related proteins", *Proc. Natl. Acad. Sci. USA* **84** (1987) 4355–4358.

[107] Anson E.L., Myers E.W., "ReAligner: A program for refining DNA sequence multialignments", in: *RECOMB'97*, 1997.

[108] Churchill G.A., Waterman M.S., "The accuracy of DNA sequences: estimating sequence quality", *Genomics* **14** (1) (1992) 89–98.

[109] Bonfield J.K., Staden R., "The application of numerical estimates of base calling accuracy to DNA sequencing projects", *Nucleic Acids Res.* **23** (8) (1995) 1406–1410.

[110] Zhou S., et al., "Whole-genome shotgun optical mapping of rhodobacter sphaeroides strain 2.4.1 and its use for whole-genome shotgun sequence assembly", *Genome Res.* **13** (9) (2003) 2142–2151.

[111] Huson D.H., Reinert K., Myers E., "The greedy path-merging algorithm for sequence assembly", in: *Proceedings of the Fifth Annual International Conference on Computational Biology (RECOMB)*, 2001.

[112] Thayer E.C., Olson M.V., Karp R.M., "Error checking and graphical representation of multiple-complete-digest (MCD) restriction-fragment maps", *Genome Res.* **9** (1) (1999) 79–90.

[113] Pop M., Kosack D., Salzberg S.L., "Hierarchical scaffolding with bambus", *Genome Res.* **14** (1) (2004) 149–159.

[114] Kim S., Liao L., Tomb J.F., "A probabilistic approach to sequence assembly validation", in: *Workshop on Data Mining in Bioinformatics*, 2001.

[115] Seto D., Koop B.F., Hood L., "An experimentally derived data set constructed for testing large-scale DNA sequence assembly algorithms", *Genomics* **15** (1993) 673–676.

[116] Miller M.J., Powell J.I., "A quantitative comparison of DNA sequence assembly programs", *J. Comp. Bio.* **1** (1994) 257–269.

[117] Huson D.H., et al., "Comparing assemblies using fragments and mate-pairs", in: *Workshop on Algorithms in Bioinformatics*, Springer-Verlag, 2001.

[118] Engle M.L., Burks C., "Artificially generated data sets for testing DNA sequence assembly algorithms", *Genomics* **16** (1993) 286–288.

[119] Myers G., "A dataset generator for whole genome shotgun sequencing", in: *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 1999, pp. 202–210.

[120] Lander E.S., et al., "Initial sequencing and analysis of the human genome", *Nature* **409** (6822) (2001) 860–921.
[121] Volik S., et al., "End-sequence profiling: Sequence-based analysis of aberrant genomes", *Proc. Natl. Acad. Sci. USA* (2003).
[122] Taillon-Miller P., et al., "Overlapping genomic sequences: a treasure trove of single-nucleotide polymorphisms", *Genome Res.* **8** (7) (1998) 748–754.
[123] Altshuler D., et al., "An SNP map of the human genome generated by reduced representation shotgun sequencing", *Nature* **407** (6803) (2000) 513–516.
[124] Mullikin J.C., et al., "An SNP map of human chromosome 22", *Nature* **407** (6803) (2000) 516–520.
[125] Dawson E., et al., "A SNP resource for human chromosome 22: extracting dense clusters of SNPs from the genomic sequence", *Genome Res.* **11** (1) (2001) 170–178.
[126] Read T.D., et al., "Comparative genome sequencing for discovery of novel polymorphisms in Bacillus anthracis", *Science* **296** (5575) (2002) 2028–2033.
[127] Lancia G., et al., "SNPs problems, complexity and algorithms", in: *9th Annual European Symposium on Algorithms (BRICS), University of Aarhus, Denmark*, 2001.
[128] Lippert R., et al., "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem", *Briefings in Bioinformatics* **3** (1) (2002) 23–31.
[129] Fasulo D., et al., "Efficiently detecting polymorphisms during the fragment assembly process", *Bioinformatics* **18** (Suppl.1) (2002) S294–S302.
[130] Casjens S., et al., "A bacterial genome in flux: the twelve linear and nine circular extrachromosomal DNAs in an infectious isolate of the Lyme disease spirochete Borrelia burgdorferi", *Mol. Microbiol.* **35** (3) (2000) 490–516.
[131] Beja O., et al., "Unsuspected diversity among marine aerobic anoxygenic phototrophs", *Nature* **415** (6872) (2002) 630–633.
[132] Randazzo C.L., et al., "Diversity, dynamics, and activity of bacterial communities during production of an artisanal Sicilian cheese as evaluated by 16S rRNA analysis", *Appl. Environ. Microbiol.* **68** (4) (2002) 1882–1892.
[133] Pearson H., "Body's bugs to be sequenced", in: *Nature Science Update*, 2003.
[134] Whitfield J., "Genome pioneer sets sights on Sargasso Sea", in: *Nature Science Update*, 2003.
[135] Kececioglu J.D., Li M., Tromp J., "Inferring a DNA sequence from erroneous copies", *Theoretical Computer Science* **185** (1) (1997) 3–13.
[136] Liang F., et al., "An optimized protocol for analysis of EST sequences", *Nucleic Acids Res.* **28** (18) (2000) 3657–3665.
[137] Kent W.J., Haussler D., *GigAssembler: An Algorithm for the Initial Assembly of the Human Genome Draft*, University of California Santa Cruz, 2000.

Advances in Large Vocabulary Continuous Speech Recognition

GEOFFREY ZWEIG AND
MICHAEL PICHENY

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
USA
gzweig@us.ibm.com
picheny@us.ibm.com

Abstract

The development of robust, accurate and efficient speech recognition systems is critical to the widespread adoption of a large number of commercial applications. These include automated customer service, broadcast news transcription and indexing, voice-activated automobile accessories, large-vocabulary voice-activated cell-phone dialing, and automated directory assistance. This article provides a review of the current state-of-the-art, and the recent research performed in pursuit of these goals.

ISSN: 0065-2458/DOI 10.1016/S0065-2458(03)60007-0

1. Introduction

Over the course of the past decade, automatic speech recognition technology has advanced to the point where a number of commercial applications are now widely deployed and successful: systems for name-dialing [84,26], travel reservations [11, 72], getting weather-information [97], accessing financial accounts [16], automated directory assistance [41], and dictation [86,9,78] are all in current use. The fact that these systems work for thousands of people on a daily basis is an impressive testimony to technological advance in this area, and it is the aim of this article to describe the technical underpinnings of these systems and the recent advances that have made them possible. It must be noted, however, that even though the technology has matured to the point of commercial usefulness, the problem of large vocabulary continuous speech recognition (LVCSR) is by no means solved: background noise, corruption by cell-phone or other transmission channels, unexpected shifts in topic, foreign accents, and overly casual speech can all cause automated systems to fail. Thus, where appropriate, we will indicate the shortcomings of current technology, and suggest areas of future research. Although this article aims for a fairly comprehensive coverage of today's speech recognition systems, a vast amount of work has been done in this area, and some limitation is necessary. Therefore, this review will focus primarily on techniques that have proven successful to the point where they have been widely adopted in competition-grade systems such as [78,36,37,58,27,93].

The cornerstone of all current state-of-the-art speech recognition systems is the Hidden Markov Model (HMM) [6,43,54,74]. In the context of HMMs, the speech recognition problem is decomposed as follows. Speech is broken into a sequence of acoustic observations or frames, each accounting for around 25 milliseconds of speech; taken together, these frames comprise the acoustics **a** associated with an utterance. The goal of the recognizer is to find the likeliest sequence of words **w** given the acoustics:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}).$$

This can then be rewritten as:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{w})P(\mathbf{a}|\mathbf{w})}{P(\mathbf{a})}.$$

Since the prior on the acoustics is independent of any specific word hypothesis, the denominator can be ignored, leaving the decomposition:

$$\arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} P(\mathbf{w})P(\mathbf{a}|\mathbf{w}).$$

The first factor, $P(\mathbf{w})$, is given by the language model, and sets the prior on word sequences. The second factor, $P(\mathbf{a}|\mathbf{w})$ is given by the acoustic model, and links word sequences to acoustics, and is described by an HMM.

The breakdown of a system into acoustic and language model components is one of the main characteristics of current LVCSR systems, and the details of these models are discussed in Sections 3 and 4. However, even with well-defined acoustic and language models that allow for the computation of $P(\mathbf{w})$ and $P(\mathbf{a}|\mathbf{w})$ for any given word and acoustic sequences $\mathbf{w}$ and $\mathbf{a}$, the problem of finding the likeliest single sequence of words remains computationally difficult, and is the subject of a number of specialized search algorithms. These are discussed in Section 5. The final component of current LVCSR systems performs the function of speaker adaptation, and adjusts the acoustic models to match the specifics of an individual voice. These techniques include Maximum A-Posteriori (MAP) adaptation [28], methods that work by adjusting the acoustic features to more closely match generic acoustic models [24], and methods that adjust the acoustic models to match the feature vectors [51]. The field of speaker adaptation has evolved quite dramatically over the past decade, and is currently a key research area; Section 6 covers it in detail. The combination of acoustic and language models, search, and adaptation that characterize current systems is illustrated in Fig. 1.

2. Front End Signal Processing

Currently, there are two main ways in which feature vectors are computed, both motivated by information about human perception. The first of these ways produces features known as *Mel Frequency Cepstral Coefficients* (MFCCs) [17], and the second method is known as *Perceptual Linear Prediction* (PLP) [38]. In both cases, the speech signal is broken into a sequence of overlapping frames which serve as the basis of all further processing. A typical frame-rate is 100 per second, with each frame having a duration of 20 to 25 milliseconds.

After extraction, the speech frames are subjected to a sequence of operations resulting in a compact representation of the perceptually important information in the

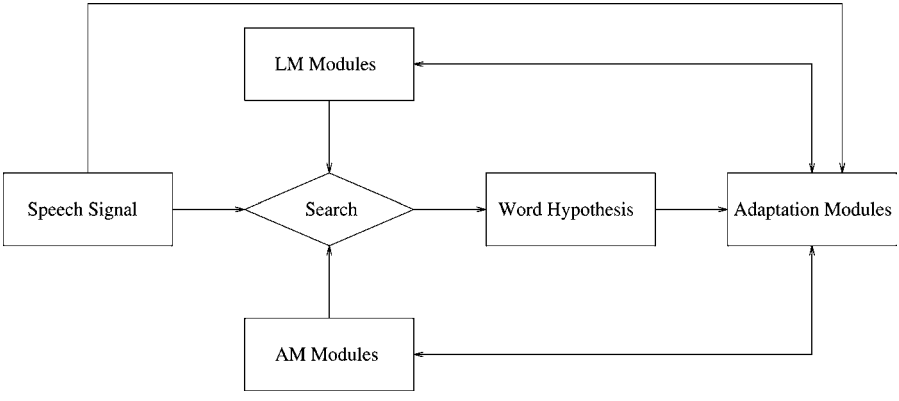


FIG. 1. Sample LVCSR architecture.

speech. Algorithmically, the steps involved in both methods are approximately the same, though the motivations and details are different. In both cases, the algorithmic process is as follows:

(1) compute the power spectrum of the frame,
(2) warp the frequency range of the spectrum so that the high-frequency range is compressed,
(3) compress the amplitude of the spectrum,
(4) decorrelate the elements of the spectral representation by performing an inverse DFT—resulting in a cepstral representation.

Empirical studies have shown that recognition performance can be further enhanced with the inclusion of features computed not just from a single frame, but from several surrounding frames as well. One way of doing this is to augment the feature vectors with the first and second temporal derivatives of the cepstral coefficients [22]. More recently, however, researchers have applied linear discriminant analysis [19] and related transforms to project a concatenated sequence of feature vectors into a low-dimensional space in which phonetic classes are well separated. The following subsections will address MFCCs, PLP features, and discriminant transforms in detail.

2.1 Mel Frequency Cepstral Coefficients

The first step in the MFCC processing of a speech frame is the computation of a short-term power spectrum [17]. In a typical application in which speech is transmitted by phone, it is sampled at 8000 Hz and bandlimited to roughly 3800 Hz. A 25 millisecond frame is typical, resulting in 200 speech samples.This is zero-padded,

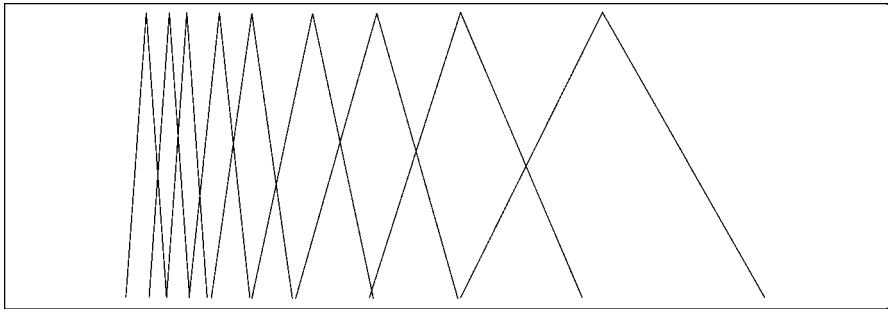


FIG. 2. Mel frequency filters grow exponentially in size.

windowed with the Hamming function

$$W(n) = 0.54 + 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$

and an FFT is used to compute a 128 point power spectrum.

The next step is to compute a warped representation of the power spectrum in which a much coarser representation is used for the high frequencies. This mirrors psychoacoustic observations that human hearing is less precise as frequency increases. To do this, the power spectrum is filtered by a sequence of triangularly shaped filterbanks, whose centers are spaced linearly on the mel scale. The Mel frequency warping [94] is given by

$$f' = 2595\log_{10}\left(1 + \frac{f}{700}\right),$$

so the bandwidth increases exponentially with frequency. Figure 2 illustrates the shape of the mel-frequency filters.[1] Typical applications use 18 to 24 filterbanks spaced between 0 and 4000 Hz [78,46]. This mel frequency warping is similar to the use of critical bands as defined in [100].

After the spectrum is passed through the mel frequency filters, the output of each filter is compressed through the application of a logarithm, and the cepstrum is computed. With F filterbank outputs x_k, the ith MFCC is given by:

$$\text{MFCC}_i = \sum_{k=1}^{F} x_k \cos\left[i\left(k - \frac{1}{2}\right)\frac{\pi}{F}\right], \quad i = 1, 2, \ldots, F.$$

In a typical implementation, the first 13 cepstral coefficients are retained.

MFCCs have the desirable property that linear channel distortions can to some extent be removed through mean subtraction. For example, an overall gain applied to

[1]The original paper [17] used fixed-width filters below 1000 Hz.

the original signal will be removed through mean-subtraction, due to the logarithmic nonlinearity. Therefore, mean-subtraction is standard.

2.2 Perceptual Linear Predictive Coefficients

Perceptual Linear Prediction is similar in implementation to MFCCs, but different in motivation and detail. In practice, these differences have proved to be important, both in lowering the overall error rate, and because PLP-based systems tend to make errors that are somewhat uncorrelated with those in MFCC systems. Therefore, as discussed later in Section 5.4, multiple systems differing in the front-end and other details can be combined through voting to reduce the error rate still further.

The principal differences between MFCC and PLP features are:

- The shape of the filterbanks.
- The use of equal-loudness preemphasis to weight the filterbank outputs.
- The use of cube-root compression rather than logarithmic compression.
- The use of a (parametric) linear-predictive model to determine cepstral coefficients, rather than the use of a (non-parametric) discrete cosine transform.

The first step in PLP analysis is the computation of a short-term spectrum, just as in MFCC analysis. The speech is then passed through a sequence of filters that are spaced at approximately one-Bark intervals, with the Bark frequency Ω being related to un-warped frequency ω (in rad/s) by:

$$\Omega(\omega) = 6\log\left\{\omega/1200\pi + \left[(\omega/1200\pi)^2 + 1\right]^{0.5}\right\}.$$

The shape of the filters is trapezoidal, rather than triangular, motivated by psychophysical experiments [79,101].

Conceptually, after the filterbank outputs are computed, they are subjected to equal-loudness preemphasis. A filterbank centered on (unwarped) frequency ω is modulated by

$$E(\omega) = \left[\left(\omega^2 + 56.8 \times 10^6\right)\omega^4\right] / \left[\left(\omega^2 + 6.3 \times 10^6\right)^2 \times \left(\omega^2 + 0.38 \times 10^9\right)\right].$$

This reflects psycho-physical experiments indicating how much energy must be present in sounds at different frequencies in order for them to be perceived as equally loud. In practice, by appropriately shaping the filters, this step can be done simultaneously with the convolution that produces their output. The weighted outputs are then cube-root compressed, $o' = o^{0.33}$.

In the final PLP step, the warped spectrum is represented with the cepstral coefficients of an all-pole linear predictive model [56]. This is similar to the DCT operation in MFCC computation, but the use of an all-pole model makes the results

more sensitive to spectral peaks, and smooths low-energy regions. In the original implementation of [38], a fifth-order autoregressive model was used; subsequent implementations use a higher order model, e.g., 12 as in [46].

2.3 Discriminative Feature Spaces

As mentioned earlier, it has been found that improved performance can be obtained by augmenting feature vectors with information from surrounding frames [22]. One relatively simple way of doing this is to compute the first and second temporal derivatives of the cepstral coefficients; in practice, this can be done by appending a number of consecutive frames (nine is typical) and multiplying with an appropriate matrix.

More recently [35,88], it has been observed that pattern recognition techniques might be applied to transform the features in a way that is more directly related to reducing the error rate. In particular, after concatenating a sequence of frames, linear discriminant analysis can be applied to find a projection that maximally separates the phonetic classes in the projected space.

Linear discriminant analysis proceeds as follows. We will denote the class associated with example i as $c(i)$. First, the means μ_j and covariances Σ_j of each class are computed, along with the overall mean μ and variance Σ:

$$\mu_j = \frac{1}{N_j} \sum_{i \text{ s.t. } c(i)=j} \mathbf{x}_i, \quad \Sigma_j = \frac{1}{N_j} \sum_{i \text{ s.t. } c(i)=j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}},$$

$$\mu = \frac{1}{N} \sum_i \mathbf{x}_i, \quad \Sigma = \frac{1}{N} \sum_i (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}.$$

Next, the total within class variance W is computed

$$W = \frac{1}{N} \sum_j N_j \Sigma_j.$$

Using θ to denote the LDA transformation matrix, the LDA objective function is given by:

$$\hat{\theta} = \arg\max_{\theta} \frac{|\theta^{\mathrm{T}} \Sigma \theta|}{|\theta^{\mathrm{T}} W \theta|},$$

and the optimal transform is given by the top eigenvectors of $W^{-1}\Sigma$.

While LDA finds a projection that tends to maximize relative interclass distances, it makes two questionable assumptions: first, that the classes are modeled by a full covariance Gaussian in the transformed space, and second that the covariances of

all transformed classes are identical. The first assumption is problematic because, as discussed in Section 3.3, full covariance Gaussians are rarely used; but the extent to which the first assumption is violated can be alleviated by applying a subsequent transformation meant to minimize the loss in likelihood between the use of full and diagonal covariance Gaussians [31]. The MLLT transform developed in [31] applies the transform ψ that minimizes

$$\sum_j N_j \left(\log\left|\mathrm{diag}\left(\psi \Sigma_j \psi^{\mathrm{T}}\right)\right| - \log\left|\psi \Sigma_j \psi^{\mathrm{T}}\right|\right)$$

and has been empirically found to be quite effective in conjunction with LDA [77].

To address the assumption of equal covariances, [77] proposes the maximization of

$$\prod_j \left(\frac{|\theta \Sigma \theta^{\mathrm{T}}|}{|\theta \Sigma_j \theta^{\mathrm{T}}|}\right)^{N_j}$$

and presents favorable results when used in combination with MLLT. A closely related technique, HLDA, [50] relates projective discriminant analysis to maximum likelihood training, where the unused dimensions are modeled with a shared covariance. This form of analysis may be used both with and without the constraint that the classes be modeled by a diagonal covariance model in the projected space, and has also been widely adopted. Combined, LDA and MLLT provide on the order of a 10% relative reduction in word-error rate [77] over simple temporal derivatives.

3. The Acoustic Model

3.1 Hidden Markov Model Framework

The job of the acoustic model is to determine word-conditioned acoustic probabilities, $P(\mathbf{a}|\mathbf{w})$. This is done through the use of Hidden Markov Models, which model speech as being produced by a speaker whose vocal tract configuration proceeds through a sequence of states, and produces one or more acoustic vectors in each state. An HMM consists of a set of states $\mathcal{S}$, a set of acoustic observation probabilities, $b_j(o)$, and a set of transition probabilities a_{ij}. The transition and observation probabilities have the following meaning:

(1) $b_j(o)$ is a function that returns the probability of generating the acoustic vector o in state j. $b_j(o_t)$ is the probability of seeing the specific acoustics associated with time t in state j. The observation probabilities are commonly modeled with Gaussian mixtures.
(2) a_{ij} is the time-invariant probability of transitioning from state i to state j.

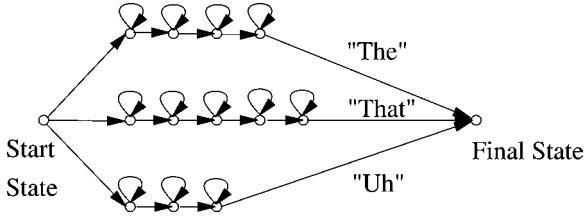


FIG. 3. A simple HMM representing the state sequence of three words. Adding an arc from the final state back to the start state would allow repetition.

Note that in the HMM framework, each acoustic vector is associated with a specific state in the HMM. Thus, a sequence of n acoustic vectors will correspond to a sequence of n consecutive states. We will denote a specific sequence of states $s_1 = a,\ s_2 = b,\ s_3 = c,\ \ldots,\ s_n = k$ by $\mathbf{s}$. In addition to normal emitting states, it is often convenient to use "null" states, which do not emit acoustic observations. In particular, we will assume that the HMM starts at time $t = 0$ in a special null start-state α, and that all paths must end in a special null final-state ω at $t = N + 1$. In general, having a specific word hypothesis $\mathbf{w}$ will be compatible with only some state sequences, $\mathbf{s}$, and not with others. It is necessary, therefore, to constrain sums over state sequences to those sequences that are compatible with a given word sequence; we will not, however, introduce special notation to make this explicit. With this background, the overall probability is factored as follows:

$$P(\mathbf{a}|\mathbf{w}) = \sum_{\mathbf{s}} P(\mathbf{a}|\mathbf{s})P(\mathbf{s}|\mathbf{w}) = \sum_{\mathbf{s}} \prod_{t=1,\ldots,n} b_{s_t}(o_t)a_{s_t s_{t-1}}.$$

Figure 3 illustrates a simple HMM that represents the state sequences of three words.

The following sections describe the components of the HMM acoustic model in more detail. Section 3.2 will focus on the mapping from words to states that is necessary to determine $P(\mathbf{s}|\mathbf{w})$. Section 3.3 discusses the Gaussian mixture models that are typically used to model $b_j(o)$. The transition probabilities can be represented in a simple table, and no further discussion is warranted. The section closes with a description of the training algorithms used for parameter estimation.

3.2 Acoustic Context Models

In its simplest form, the mapping from words to states can be made through the use of a phonetic lexicon that associates one or more sequences of phonemes with each word in the vocabulary. For example,

```
barge    |     B AA R JH
tomato   |     T AH M EY T OW
tomato   |     T AH M AA T OW
```

Typically, a set of 40 phonemes is used, and comprehensive dictionaries are available [14,15].

In practice, coarticulation between phones causes this sort of invariant mapping to perform poorly, and instead some sort of context-dependent mapping from words to acoustic units is used [95,5]. This mapping takes each phoneme and the phonemes that surround it, and maps it into an acoustic unit. Thus, the "AA" in "B AA R JH" may have a different acoustic model than the "AA" in "T AH M AA T OW." Similarly, the "h" in "hammer" may be modeled with a different acoustic unit depending on whether it is seen in the context of "the hammer" or "a hammer." The exact amount of context that is used can vary, the following being frequently used:

(1) Word-internal triphones. A phone and its immediate neighbors to the left and right. However, special units are used at the beginnings and endings of words so that context does not persist across word boundaries.
(2) Cross-word triphones. The same as above, except that context persists across word boundaries, resulting in better coarticulation modeling.
(3) Cross-word quinphones. A phone and its two neighbors to the left and right.
(4) A phone, and all the other phones in the same word.
(5) A phone, all the other phones in the same word, and all phones in the preceding word.

When a significant amount of context is used, the number of potential acoustic states becomes quite large. For example, with triphones the total number of possible acoustic models becomes approximately $40^3 = 64{,}000$. In order to reduce this number, decision-tree clustering is used to determine equivalence classes of phonetic contexts [5,95]. A sample tree is shown in Fig. 4. The tree is grown in a top-down fashion using an algorithm similar to that of Fig. 5. Thresholds on likelihood gain, frame-counts, or the Bayesian information criterion [10] can be used to determine an appropriate tree depth.

In a typical large vocabulary recognition system [78], it is customary to have a vocabulary size between 30 and 60 thousand words and two or three hundred hours of training data from hundreds of speakers. The resulting decision trees typically have between 4000 and 12,000 acoustic units [78,46].

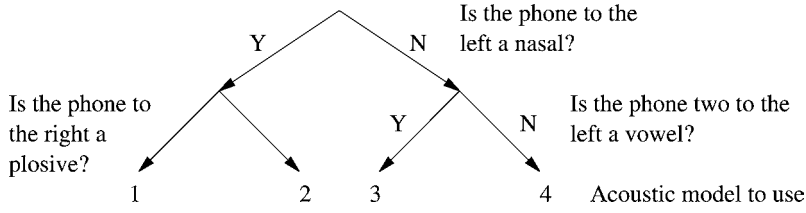


FIG. 4. Decision tree for clustering phonetic contexts.

1. Create a record for each frame that includes the frame and the phonetic context associated with it.
2. Model the frames associated with a node with a single diagonal-covariance Gaussian. The frames associated with a node will have a likelihood according to this model.
3. For each yes/no question based on the context window, compute the likelihood that would result from partitioning the examples according to the induced split.
4. Split the frames in the node using the question that results in the greatest likelihood gain, and recursively process the resulting two nodes.

FIG. 5. Decision tree building.

3.3 Gaussian Mixture State Models

The observation probabilities $b_j(o)$ are most often modeled with mixtures of Gaussians. The likelihood of the d-dimensional feature vector $\mathbf{x}$ being emitted by state j is given by:

$$b_j(\mathbf{x}) = \sum_k m_{jk}\left((2\pi)^d|\Sigma_{jk}|\right)^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{jk})^{\mathrm{T}}\Sigma_{jk}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{jk})\right)$$

where the coefficients m_{jk} are mixture weights, $\sum_k m_{jk} = 1$. This can be expressed more compactly as

$$b_j(\mathbf{x}) = \sum_k m_{jk}\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jk}, \Sigma_{jk}).$$

In order to minimize the amount of computation required to compute observation probabilities, it is common practice to use diagonal covariance matrices. Between 150,000 and 300,000 Gaussians are typical in current LVCSR systems.

The use of diagonal covariance matrices has proved adequate, but requires that the dimensions of the feature vectors be relatively uncorrelated. While the linear transforms described in Section 2 can be used to do this, recently there has been a significant amount of work focused on more efficient covariance representations.

One example of this is EMLLT [70], in which the inverse covariance matrix of each Gaussian j is modeled as the sum of basis matrices. First, a set of d dimensional basis vectors a_l is defined. Then inverse covariances are modeled as:

$$\Sigma_j^{-1} = \sum_{l=1}^{D} \lambda_l^j \mathbf{a}_l \mathbf{a}_l^{\mathrm{T}}.$$

One of the main contributions of [70] is to describe a maximum-likelihood training procedure for adjusting the basis vectors. Experimental results are presented that show improved performance over both diagonal and full-covariance modeling in a recognition system for in-car commands. In further work [3], this model has been generalized to model both means and inverse-covariance matrices in terms of basis expansions (SPAM).

3.4 Maximum Likelihood Training

A principal advantage of HMM-based systems is that it is quite straightforward to perform maximum likelihood parameter estimation. The main step is to compute posterior state-occupancy probabilities for the HMM states. To do this, the following quantities are defined:

- $\alpha_j(t)$: the probability of the observation sequence up to time t, and accounting for o_t in state j.
- $\beta_j(t)$: the probability of the observation sequence $o_{t+1} \ldots o_N$ given that the state at time t is j.
- $P = \sum_k \alpha_k(t)\beta_k(t)$: the total data likelihood, constant over t.
- $\gamma_j(t) = \frac{\alpha_j(t)\beta_j(t)}{\sum_k \alpha_k(t)\beta_k(t)}$: the posterior probability of being in state j at time t.
- $\mathrm{mix}_{jk}(t) = \frac{m_{jk}\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})}{\sum_f m_{jf}\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jf}, \boldsymbol{\Sigma}_{jf})}$: the probability of mixture component k given state j at time t.

The α and β quantities can be computed with a simple recursion:

- $\alpha_j(t) = \sum_i \alpha_i(t-1)a_{ij}b_j(o_t)$.
- $\beta_j(t) = \sum_k a_{jk}b_k(o_{t+1})\beta_k(t+1)$.

The recursions are initialized by setting all αs and βs to 0 except:

- $\alpha_\alpha(0) = 1$.
- $\beta_\omega(N+1) = 1$.

Once the posterior state occupancy probabilities are computed, it is straightforward to update the model parameters for a diagonal-covariance system [54,94,73].

- $\hat{a}_{ij} = \frac{\sum_t \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)}{\sum_t \alpha_i(t)\beta_i(t)}$.
- $\hat{\mu}_{jk} = \frac{\sum_t \gamma_j(t) \text{mix}_{jk}(t) \mathbf{o}_t}{\sum_t \gamma_j(t) \text{mix}_{jk}(t)}$.
- $\hat{\Sigma}_{jk} = \frac{\sum_t \gamma_j(t) \text{mix}_{jk}(t) (\mathbf{o}_t - \hat{\mu}_{jk})(\mathbf{o}_t - \hat{\mu}_{jk})^{\mathrm{T}}}{\sum_t \gamma_j(t) \text{mix}_{jk}(t)}$.

This discussion has avoided a number of subtleties that arise in practice, but are not central to the ideas. Specifically, when multiple observation streams are available, an extra summation must be added outside all others in the reestimation formulae. Also, observation probabilities are tied across multiple states—the same "ae" acoustic model may be used in multiple HMM states. This entails creating summary statistics for each acoustic model by summing the statistics of all the states that use it. Finally, in HMMs with extensive null states, the recursions and reestimation formulae must be modified to reflect the spontaneous propagation of probabilities through chains of null states.

3.4.1 Maximum Mutual Information Training

In standard maximum likelihood training, the model parameters for each class are adjusted in isolation, so as to maximize the likelihood of the examples of that particular class. While this approach is optimal in the limit of infinite training data [62], it has been suggested [63,4] that under more realistic conditions, a better training objective might be to maximize the amount of mutual information between the acoustic vectors and the word labels. That is, rather than training so as to maximize

$$P_\theta(\mathbf{w}, \mathbf{a}) = P_\theta(\mathbf{w}) P_\theta(\mathbf{a}|\mathbf{w})$$

with respect to θ, to train so as to maximize

$$\sum_{\mathbf{w},\mathbf{a}} P_\theta(\mathbf{w}, \mathbf{a}) \log \frac{P_\theta(\mathbf{w}, \mathbf{a})}{P_\theta(\mathbf{w}) P_\theta(\mathbf{a})}.$$

Using the training data D to approximate the sum over all words and acoustics, we can represent the mutual information as

$$\sum_D \log \frac{P_\theta(\mathbf{a}, \mathbf{w})}{P_\theta(\mathbf{w}) P_\theta(\mathbf{a})} = \sum_D \log \frac{P_\theta(\mathbf{w}) P_\theta(\mathbf{a}|\mathbf{w})}{P_\theta(\mathbf{w}) P_\theta(\mathbf{a})} = \sum_D \log \frac{P_\theta(\mathbf{a}|\mathbf{w})}{P_\theta(\mathbf{a})}$$

$$= \sum_D \log \frac{P_\theta(\mathbf{a}|\mathbf{w})}{\sum_{\mathbf{w}'} P_\theta(\mathbf{w}') P_\theta(\mathbf{a}|\mathbf{w}')}.$$

If we assume that the language model determining $P_\theta(\mathbf{w})$ is constant (as is the case in acoustic model training) then this is identical to optimizing the posterior word probability:

$$\sum_D \log P_\theta(\mathbf{w}|\mathbf{a}) = \sum_D \log \frac{P_\theta(\mathbf{a}|\mathbf{w})P_\theta(\mathbf{w})}{\sum_{\mathbf{w}'} P_\theta(\mathbf{w}')P_\theta(\mathbf{a}|\mathbf{w}')}.$$

Before describing MMI training in detail, we note that the procedure that will emerge is not much different from training an ML system. Procedurally, one first computes the state-occupancy probabilities and first and second order statistics exactly as for a ML system. This involves summing path posteriors over all HMM paths that are consistent with the known word hypotheses. One then repeats exactly the same process, but sums over all HMM paths without regard to the transcripts. The two sets of statistics are then combined in a simple update procedure. For historical reasons, the first set of statistics is referred to as "numerator" statistics and the second (unconstrained) set as "denominator" statistics.

An effective method for performing MMI optimization was first developed in [30] for the case of discrete hidden Markov models. The procedure of [30] works in general to improve objective functions $R(\theta)$ that are expressible as

$$R(\theta) = \frac{s_1(\theta)}{s_2(\theta)}$$

with s_1 and s_2 being polynomials with $s_2 > 0$. Further, for each individual probability distribution λ under adjustment, it must be the case that $\lambda_i \geqslant 0$ and $\sum_i \lambda_i = 1$. In this case, it is proved that the parameter update

$$\hat{\lambda}_i = \frac{\lambda_i\left(\frac{\partial \log R(\lambda)}{\partial \lambda_i} + D\right)}{\sum_k \lambda_k\left(\frac{\partial \log R(\lambda)}{\partial \lambda_k} + D\right)}$$

is guaranteed to increase the objective function, with a large enough value of the constant D. In the case of discrete variables, it is shown that

$$\frac{\partial \log R(\lambda)}{\partial \lambda_i} = \frac{1}{\lambda_i}\left(C_{\lambda_i}^{\text{num}} - C_{\lambda_i}^{\text{den}}\right)$$

where λ_i is probability of event associated with λ_i being true, and C_{λ_i} is count of times this event occurred, as computed from the α–β recursions of the previous section.

Later work [68,92], extended these updates to Gaussian means and variances, and [92] did extensive work to determine appropriate values of D for large vocabulary speech recognition. For state j, mixture component m, let $S(x)$ denote the first order statistics, $S(x^2)$ denote the second order statistics, and C denote the count of the

number of times a mixture component is used. The update derived is

$$\hat{\mu}_{jm} = \frac{S_{jm}^{\text{num}}(x) - S_{jm}^{\text{den}}(x) + D\mu_{jm}}{C_{jm}^{\text{num}} - C_{jm}^{\text{den}} + D},$$

$$\hat{\sigma}_{jm}^2 = \frac{S_{jm}^{\text{num}}(x^2) - S_{jm}^{\text{den}}(x^2) + D(\sigma_{jm}^2 + \mu_{jm}^2)}{C_{jm}^{\text{num}} - C_{jm}^{\text{den}} + D} - \hat{\mu}_{jm}^2.$$

For the mixture weights, let f_{jm} be the mixture coefficient associated with mixture component m of state j. Then

$$\hat{f}_{jm} = \frac{f_{jm}\left(\frac{\partial \log R(\lambda)}{\partial f_{jm}} + D\right)}{\sum_k f_{jk}\left(\frac{\partial \log R(\lambda)}{\partial f_{jk}} + D\right)}$$

with

$$\frac{\partial \log R(\lambda)}{\partial f_{jk}} = \frac{1}{f_{jk}}\left(C_{jk}^{\text{num}} - C_{jk}^{\text{den}}\right).$$

Several alternative ways for reestimating the mixture weights are given in [92].

MMI has been found to give a 5–10% relative improvement in large vocabulary tasks [92], though the advantage diminishes as systems with larger numbers of Gaussians are used [58]. The main disadvantage of MMI training is that the denominator statistics must be computed over all possible paths. This requires either doing a full decoding of the training data at each iteration, or the computation of lattices (see Section 5.2). Both options are computationally expensive unless an efficiently written decoder is available.

4. Language Model

4.1 Finite State Grammars

Finite state grammars [1,39] are the simplest and in many ways the most convenient way of expressing a language model for speech recognition. The most basic way of expressing one of these grammars is as an unweighted regular expression that represents a finite set of recognizable statements. For example, introductions to phone calls in a three-person company might be represented with the expression

```
(Hello | Hi) (John | Sally | Sam)? it's
(John | Sally | Sam)
```

At a slightly higher level, Backus Naur Form [64] is often used for more elaborate grammars with replacement patterns. For example,

```
<SENTENCE> ::= Greeting.
Greeting ::= Intro Name? it's Name.
Intro ::= Hello | Hi.
Name ::= John | Sally | Sam.
```

In fact, BNF is able to represent context free grammars [13]—a broad class of grammars in which recursive rule definitions allow the recognition of some strings that cannot be represented with regular expressions. However, in comparison with regular expressions, context-free grammars have had relatively little affect on ASR, and will not be discussed further.

Many of the tools and conventions associated with regular expressions were developed in the context of computer language compilers, in which texts (programs) were either syntactically correct or not. In this context, there is no need for a notion of how correct a string is, or alternatively what the probability of it being generated by a speaker of the language is. Recall, however, that in the context of ASR, we are interested in $P(\mathbf{w})$, the probability of a word sequence. This can easily be incorporated in to the regular expression framework, simply by assigning costs or probabilities to the rules in the grammar.

Grammars are frequently used in practical dialog applications, where developers have the freedom to design system prompts and then specify a grammar that is expected to handle all reasonable replies. For example, in an airline-reservation application the system might ask "Where do you want to fly to?" and then activate a grammar designed to recognize city names. Due to their simplicity and intuitive nature, these sorts of grammars are the first choice wherever possible.

4.2 N-gram Models

N-gram language models are currently the most widely used LMs in large vocabulary speech recognition. In an N-gram language model, the probability of each word is conditioned on the $n-1$ preceding words:

$$P(\mathbf{w}) = P(w_1)P(w_2|w_1)\cdots P(w_{n-1}|w_1\ldots w_{n-2}) \times \prod_{i=n}^{i=N} P(w_i|w_{i-1}, w_{i-2}, \ldots, w_{i-n+1}).$$

While in principle this model ignores a vast amount of prior knowledge concerning linguistic structure—part-of-speech classes, syntactic constraints, semantic coher-

ence, and pragmatic relevance—in practice, researchers have been unable to significantly improve on it.

A typical large vocabulary system will recognize between 30 and 60 thousand words, and use a 3 or 4-gram language model trained on around 200 million words [78]. While 200 million words seems at first to be quite large, in fact for a 3-gram LM with a 30,000 word vocabulary, it is actually quite small compared to the 27×10^{12} distinct trigrams that need to be represented. In order to deal with this problem of data sparsity, a great deal of effort has been spent of developing techniques for reliably estimating the probabilities of rare events.

4.2.1 Smoothing

Smoothing is perhaps the most important practical detail in building N-gram language models, and these techniques fall broadly into three categories: additive smoothing, backoff models, and interpolated models. The following sections touch briefly on each, giving a full description for only interpolated LMs, which have been empirically found to give good performance on a variety of tasks. The interested reader can find a full review of all these methods in [12].

4.2.1.1 Additive Smoothing. In the following, we will use the compact notation w_x^y to refer to the sequence of words $w_x, w_{x+1} \ldots w_y$, and $c(w_x^y)$ to the number of times (count) that this sequence has been seen in the training data. The maximum-likelihood estimate of $P(w_i|w_{i-n+1}^{i-1})$ is thus given as:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}.$$

The problem, of course, is that for high-order N-gram models, many of the possible (and perfectly normal) word sequences in a language will not be seen, and thus assigned zero-probability. This is extraordinarily harmful to a speech recognition system, as one that uses such a model will never be able to decode these novel word sequences. One of the simplest ways of dealing with such a problem is to use a set of fictitious or imaginary counts to encode our prior knowledge that all word sequences have some likelihood. In the most basic implementation [42], one simply adds a constant amount δ to each possible event. For a vocabulary of size $|V|$, one then has:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta|V| + c(w_{i-n+1}^{i-1})}.$$

The optimal value of δ can be found simply by performing a search so as to maximize the implied likelihood on a set of held-out data. This scheme, while having the virtue of simplicity, tends to perform badly in practice [12].

4.2.1.2 Low-Order Backoff. One of the problems of additive smoothing is that it will assign the same probability to all unseen words that follow a particular history. Thus for example, it will assign the same probability to the sequence "spaghetti western" as to "spaghetti hypanthium," assuming that neither has been seen in the training data. This violates our prior knowledge that more frequently occurring words are more likely to occur, even in previously unseen contexts.

One way of dealing with this problem is to use a backoff model in which one "backs off" to a low order language model estimate to model unseen events. These models are of the form:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \begin{cases} \alpha(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0, \\ \gamma(w_{i-n+1}^{i-1})P(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0. \end{cases}$$

One example of this is Katz smoothing [45], which is used, e.g., in the SRI language-modeling toolkit [83]. However, empirical studies have shown that better smoothing techniques exist, so we will not present it in detail.

4.2.1.3 Low-Order Interpolation. The weakness of a backoff language model is that it ignores the low-order language model estimate whenever a high-order N-gram has been seen. This can lead to anomalies when some high-order N-grams are seen, and others with equal (true) probability are not. The most effective type of N-gram model uses an interpolation between high and low-order estimates under all conditions. Empirically, the most effective of these is the modified Kneser–Ney language model [12], which is based on [47].

This model makes use of the concept of the number of unique words that have been observed to follow a given language model history at least k times. Define

$$N_k\left(w_{i-n+1}^{i-1}\bullet\right) = \left|\left\{w_i\colon c\left(w_{i-n+1}^{i-1}w_i\right) = k\right\}\right|$$

and

$$N_{k+}\left(w_{i-n+1}^{i-1}\bullet\right) = \left|\left\{w_i\colon c\left(w_{i-n+1}^{i-1}w_i\right) \geqslant k\right\}\right|.$$

The modified Kneser Ney estimate is then given as

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{c(w_{i-n+1}^{i-1})} + \gamma\left(w_{i-n+1}^{i-1}\right)P\left(w_i|w_{i-n+2}^{i-1}\right).$$

Defining

$$Y = \frac{n_1}{n_1 + 2n_2}$$

where n_r is the number of n-grams that occur exactly r times, the discounting factors are given by

$$D(c) = \begin{cases} 0 & \text{if } c = 0, \\ 1 - 2Y\frac{n_2}{n_1} & \text{if } c = 1, \\ 2 - 3Y\frac{n_3}{n_2} & \text{if } c = 2, \\ 3 - 4Y\frac{n_4}{n_3} & \text{if } c \geqslant 3. \end{cases}$$

The backoff weights are determined by

$$\gamma\left(w_{i-n+1}^{i-1}\right) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{c(w_{i-n+1}^{i-1})}.$$

This model has been found to slightly outperform most other models and is in use in state-of-the-art systems [78]. Because $D(0) = 0$, this can also be expressed in a backoff form.

4.2.2 Cross-LM Interpolation

In many cases, several disparate sources of language model training data are available, and the question arises: what is the best method of combining these sources? The obvious answer is simply to concatenate all the sources of training data together, and to build a model. This, however, has some serious drawbacks when the sources are quite different in size. For example, in many systems used to transcribe telephone conversations [78,76,93,27], data from television broadcasts is combined with a set of transcribed phone conversations. However, due to its easy availability, there is much more broadcast data than conversational data: about 150 million words compared to 3 million. This can have quite negative effects. For example, in the news broadcast data, the number of times "news" follows the bigram "in the" may be quite high, whereas in conversations, trigrams like "in the car" or "in the office" are much likelier. Because of the smaller amount of data, though, these counts will be completely dwarfed by the broadcast news counts, with the result that the final language model will be essentially identical to the broadcast news model. Put another way, it is often the case that training data for several *styles of speaking* is available, and that the relative amounts of data in each category bears no relationship to how frequently the different styles are expected to be used in real life.

In order to deal with this, it is common to interpolate multiple distinct language models. For each data source k, a separate language model is built that predicts word

probabilities: $P_k(w_i|w_{i-n+1}^{i-1})$. These models are then combined with weighting factors λ_k:

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_k P_k\left(w_i|w_{i-n+1}^{i-1}\right), \quad \sum_k \lambda_k = 1.$$

For example, in a recent conversational telephony system [78] an interpolation of data gathered from the web, broadcast news data, and two sources of conversational data (with weighting factors 0.4, 0.2, 0.2, and 0.2 respectively) resulted in about a 10% relative improvement over using the largest single source of conversational training data.

4.2.3 *N*-gram Models as Finite State Graphs

While N-gram models have traditionally been treated as distinct from recognition grammars, in fact they are identical, and this fact has been increasingly exploited. One simple way of seeing this is to consider a concrete algorithm for constructing a finite state graph at the HMM state level from an N-gram language model expressed as a backoff language model. This will make use of two functions that act on a word sequence w_j^k:

(1) head(w_j^k) returns the suffix w_{j+1}^k.
(2) tail(w_j^k) returns the prefix w_j^{k-1}.

For a state-of-the-art backoff model, one proceeds as follows:

(1) for each N-gram with history $\mathbf{q}$ and successor word r make a unique state for $\mathbf{q}$, head($\mathbf{q}r$), and tail($\mathbf{q}$),
(2) for each N-gram add an arc from $\mathbf{q}$ to head($\mathbf{q}r$) labeled with r and weighted by the α probability of the backoff model,
(3) for each unique N-gram history $\mathbf{q}$ add an arc from $\mathbf{q}$ to tail($\mathbf{q}$) with the backoff γ associated with $\mathbf{q}$.

To accommodate multiple pronunciations of a given word, one then replaces each word arc with a set of arcs, one labeled with each distinct pronunciation, and multiplies the associated probability with the probability of that pronunciation. For acoustic models in which there is no cross word context, each pronunciation can then be replaced with the actual sequence of HMM states associated with the word; accommodating cross word context is more complex, but see, e.g., [99]. Figure 6 illustrates a portion of an HMM n-gram graph.

We have described the process of expanding a language model into a finite-state graph as a sequence of "search and replace" operations acting on a basic representation at the word level. However, [59,60] have recently argued that the process is

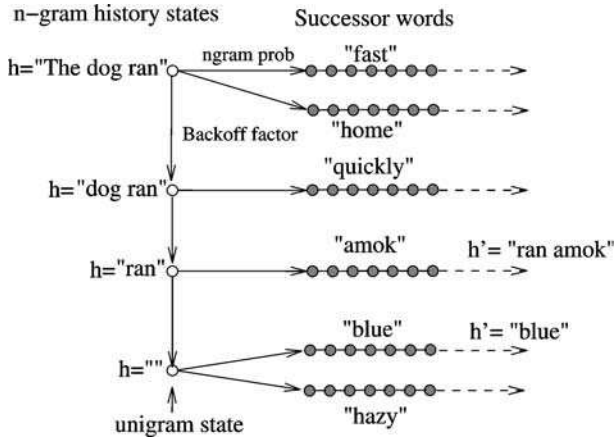


FIG. 6. HMM state graph illustrating the structure of a backoff language model.

best viewed in terms of a sequence of finite state transductions. In this model, one begins with a finite state encoding of the language model, but represents the expansion at each level—from word to pronunciation, pronunciation to phone, and phone to state—as the composition of the previous representation with a finite state transducer. The potential advantage of this approach is a consistent representation of each form of expansion, with the actual operations being performed by a single composition function. In practice, care must be taken to ensure that the composition operations do not use large amounts of memory, and in some cases, it is inconvenient to express the acoustic context model in the form of a transducer (e.g., when long span context models are used).

In some ways, the most important advantage of finite-state representations is that operations of determinization and minimization were recently developed by [59,60]. Classical algorithms were developed in the 1970s [1] for unweighted graphs as found in compilers, but the extension to weighted graphs (the weights being the language model and transition probabilities) has made these techniques relevant to speech recognition. While it is beyond the scope of this paper to present the algorithms for determinization and minimization, we briefly describe the properties.

A graph is said to be deterministic if each outgoing arc from a given state has a unique label. In the context of speech recognition graphs, the arcs are labeled with either HMM states, or word, pronunciation, or phone labels. While the process of taking a graph and finding an equivalent deterministic one is well defined, the deterministic representation can in pathological cases grow exponentially in the number of states of the input graph. In practice, this rarely happens, but the graph does

grow. The benefit actually derives from the specific procedures used to implement the Viterbi search described in Section 5.1. Suppose one has identified a fixed number w of states that are reasonably likely at a given time t. Only a small number k of HMM states are likely to have good acoustic matches, and thus to lead to likely states at time $t+1$. Thus, if on average z outgoing arcs per state are labeled with a given HMM state, the number of likely states at $t+1$ will be on the order of zkw. By using a deterministic graph, z is limited to 1, and thus tends to decrease the number of states that will ever be deemed likely. In practice, this property can lead to an order-of-magnitude speedup in search time, and makes determinization critical.

One can also ask, given a deterministic graph, what is the smallest equivalent deterministic graph. The process of minimization [59] produces such a graph, and in practice often reduces graph sizes by a factor of two or three.

4.2.4 Pruning

Modern corpus collections [33] often contain an extremely large amount of data—between 100 million and a billion words. Given that N-gram language models can backoff to lower-order statistics when high-order statistics are unavailable, and that representing extremely large language models can be disadvantageous from the point-of-view of speed and efficiency, it is natural to ask how one can trade off language model size and fidelity. Probably the simplest way of doing this is to impose a count threshold, and then to use a lower-order backoff estimate for the probability of the nth word in such N-grams.

A somewhat more sophisticated approach [80] looks at the loss in likelihood caused by using the backoff estimate to select N-grams to prune. Using P and P' to denote the original and backed-off estimates, and $N(\cdot)$ to represent the (possibly discounted) number of times an N-gram occurs, the loss in log likelihood caused by the omission of an N-gram w_{i-n+1}^{i} is given by:

$$N\left(w_{i-n+1}^{i}\right)\left(\log P\left(w_i|w_{i-n+1}^{i-1}\right) - \log P\left(w_i|w_{i-n+2}^{i-1}\right)\right).$$

In the "Weighted Difference Method" [80], one computes all these differences, and removes the N-grams whose difference falls below a threshold. A related approach [82] uses the Kullback–Leibler distance between the original and pruned language models to decide which N-grams to prune. The contribution of an N-gram in the original model to this KL distance is given by:

$$P\left(w_{i-n+1}^{i}\right)\left(\log P\left(w_i|w_{i-n+1}^{i-1}\right) - \log P\left(w_i|w_{i-n+2}^{i-1}\right)\right)$$

and the total KL distance is found by summing over all N-grams in the original model. The algorithm of [82] works in batch mode, first computing the change in relative entropy that would result from removing each N-gram, and then removing

all those below a threshold, and recomputing backoff weights. A comparison of the weighted-difference and relative-entropy approaches shows that the two criteria are the same in form, and the difference between the two approaches is primarily in the recomputation of backoff weights that is done in [82]. In practice, LM pruning can be extremely useful in limiting the size of a language model in compute-intensive tasks.

4.2.5 Class Language Models

While n-gram language models often work well, they have some obvious drawbacks, specifically their inability to capture linguistic generalizations. For example, if one knows that the sentence "I went home to feed my dog" has a certain probability, then one might also surmise that the sentence "I went home to feed my cat" is also well-formed, and should have roughly the same probability. There are at least two forms of knowledge that are brought to bear to make this sort of generalization: syntactic and semantic. Syntactically, both "dog" and "cat" are nouns, and can therefore be expected to be used in the same ways in the same sentence patterns. Further, we have the semantic information that both are pets, and this further strengthens their similarity. The importance of the semantic component can be further highlighted by considering the two sentences, "I went home to walk my dog," and "I went home to walk my cat." Here, although the syntactic structure is the same, the second sentence seems odd because cats are not walked.

Class-based language models are an attempt to capture the syntactic generalizations that are inherent in language. The basic idea is to first express a probability distribution over parts-of-speech (nouns, verbs, pronouns, etc.), and then to specify the probabilities of specific instances of the parts of speech. In its simplest form [8] a class based language model postulates that each word maps to a single class, so that the word stream w_i^k induces a sequence of class labels c_i^k. The n-gram word probability is then given by:

$$P\left(w_i | w_{i-n+1}^{i-1}\right) = P(w_i | c_i) P\left(c_i | c_{i-n+1}^{i-1}\right).$$

Operationally, one builds an n-gram model on word classes, and then combines this with a unigram model that specifies the probability of a specific word given a class. This form of model makes the critical assumption that each word maps into a unique class, which of course is not true for standard parts of speech. (For example, "fly" has a meaning both in the verb sense of what a bird does, and in the noun sense of an insect.) However, [8] present an automatic procedure for learning word-classes of this form. This method greedily assigns words to classes so as to minimize the perplexity of induced N-gram model over class sequences. This has the advantage both of relieving the user from specifying grammatical relationships, and of being

able to combine syntactic and semantic information. For example, [8] presents a class composed of *feet miles pounds degrees inches barrels tons acres meters bytes* and many similar classes whose members are similar both syntactically and semantically.

Later work [66] extends the class-based model to the case where a word may map into multiple classes, and a general mapping function $S(\cdot)$ is used to map a word history w_{i-n+1}^{i-1} into a specific equivalence class s. Under these more general assumptions, we have

$$P\left(w_i|w_{i-n+1}^{i-1}\right) = \sum_{c_i} P(w_i|c_i)\left[\sum_s P(c_i|s)P\left(s|w_{i-n+1}^{i-1}\right)\right].$$

Due to the complexity of identifying reasonable word-to-class mappings, however, the class induction procedure presented assumes an unambiguous mapping for each word.

This general approach has been further studied in [67], and experimental results are presented suggesting that automatically derived class labels are superior to the use of linguistic part-of-speech labels. The process can also be simplified [91] to using

$$P\left(w_i|c\left(w_{i-n+1}^{i-1}\right)\right).$$

Class language models are now commonly used in state-of-the-art systems, where their probabilities are interpolated with word-based N-gram probabilities, e.g., [93].

5. Search

Recall that the objective of a decoder is to find the best word sequence $\mathbf{w}^*$ given the acoustics:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{w})P(\mathbf{a}|\mathbf{w})}{P(\mathbf{a})}.$$

The crux of this problem is that with a vocabulary size V and utterance length N, the number of possible word-sequences is $\mathrm{O}(V^N)$, i.e., it grows exponentially in the utterance length. Over the years, the process of finding this word sequence has been one of the most studied aspects of speech recognition with numerous techniques and variations developed, [29,69,2]. Interestingly, in recent years, there has been a renaissance of interest in the simplest of these decoding algorithms: the Viterbi procedure. The development of better HMM compilation techniques along with faster computers has made Viterbi applicable to both large vocabulary recognition and constrained tasks, and therefore this section will focus on Viterbi alone.

5.1 The Viterbi Algorithm

The Viterbi algorithm operates on an HMM graph in order to find the best alignment of a sequence of acoustic frames to the states in the graph. For the purposes of this discussion, we will define an HMM in the classical sense as consisting of states with associated acoustic models, and arcs with associated transition costs. A special non-emitting "start state" α and "final state" ω are specified such that all paths start at $t = 0$ in α and end at $t = N + 1$ in ω. Finally, we will associate a string label (possibly "epsilon" or *null*) with each arc. The semantics of Viterbi decoding can then be very simply stated: the single best alignment of the frames to the states is identified, and the word labels encountered on the arcs of this path are output. Note that in the "straight" HMM framework there is no longer any distinction between acoustic model costs, language model costs, or any other costs. All costs associated with all sources of information must be incorporated in the transition and emission costs that define the network: $b_j(o_t)$ and A_{ij}.

A more precise statement of Viterbi decoding is to find the optimal state sequence $\mathbf{S}^* = s_1, s_2, \ldots, s_N$:

$$\mathbf{S}^* = \arg\max_{\mathbf{S}} \prod_{t=1,\ldots,n} b_{s_t}(o_t) a_{s_t s_{t-1}}.$$

Remarkably, due to the limited-history property of HMMs, this can be done with an extremely simple algorithm [54,73]. We define

(1) $\delta_t(j)$: the cost of the best path ending in state j at time t,
(2) $\Psi_t(j)$: the state preceding state j on the best path ending in state t at time t,
(3) pred(s): the set of states that are s's immediate predecessors in the HMM graph.

These quantities can then be computed for all states and all times according to the recursions

(1) *Initialize*

- $\delta_0(\alpha) = 1$,
- $\Psi_0(s) =$ undefined $\forall s$,
- $\delta_0(s) = 0\ \forall s \neq \alpha$;

(2) *Recursion*

- $\delta_t(s) = \max_{j \in \text{pred}(s)} \delta_{t-1}(j) A_{js} b_t(s)$,
- $\Psi_t(s) = \arg\max_{j \in \text{pred}(s)} \delta_{t-1}(j) A_{js} b_t(s)$.

Thus, to perform decoding, one computes the δs and their backpointers Ψ, and then follows the backpointers backwards from the final state ω at time $N+1$. This produces the best path, from which the arc labels can be read off.

In practice, there are a several issues that must be addressed. The simplest of these is that the products of probabilities that define the δs will quickly underflow arithmetic precision. This can be easily dealt with by representing numbers with their logarithms instead. A more difficult issue occurs when non-emitting states are present throughout the graph. The semantics of null states in this case are that spontaneous transitions are allowed without consuming any acoustic frames. The update for a given time frame must then proceed in two stages:

(1) The δs for emitting states are computed in any order by looking at their predecessors.
(2) The δs for null states are computed by iterating over them in topological order and looking at their predecessors.

The final practical issue is that in large systems, it may be advantageous to use pruning to limit the number of states that are examined at each time frame. In this case, one can maintain a fixed number of "live" states at each time frame. The decoding must then be modified to "push" the δs of the live states at time t to the *successor* states at time $t+1$.

An examination of the Viterbi recursions reveals that for an HMM with A arcs and an utterance of N frames, the runtime is $O(NA)$ and the space required is $O(NS)$. However, it is interesting to note that through the use of a divide-and-conquer recursion, the space used can be reduced to $O(Sk \log_k N)$ at the expense of a runtime of $O(NA \log_k N)$ [98]. This is often useful for processing long conversations, messages or broadcasts. The Viterbi algorithm can be applied to any HMM, and the primary distinction is whether the HMM is explicitly represented and stored in advance, or whether it is constructed "on-the-fly." The following two sections address these approaches.

5.1.1 Statically Compiled Decoding Graphs (HMMs)

Section 4.2.3 illustrated the conversion of an N-gram based language model into a statically compiled HMM, and in terms of decoding efficiency, this is probably the best possible strategy [60,78]. In this case, a large number of optimizations can be applied to the decoding graph [60] at "compile time" so that a highly efficient representation is available at decoding time without further processing. Further, it provides a unified way of treating both large and small vocabulary recognition tasks.

5.1.2 Dynamically Compiled Decoding Graphs (HMMs)

Unfortunately, under some circumstances it is difficult or impossible to statically represent the search space. For example, in a cache-LM [48,49] one increases the probability of recently spoken words. Since it is impossible to know what will be said at compile-time, this is poorly suited to static compilation. Another example is the use of *trigger-LMs* [75] in which the co-occurrences of words appearing throughout a sentence are used to determine its probability; in this case, the use of a long-range word-history makes graph compilation difficult. Or in a dialog application, one may want to create a grammar that is specialized to information that a user has just provided; obviously, this cannot be anticipated at compile time. Therefore, despite its renaissance, the use of static decoding graphs is unlikely to become ubiquitous.

In the cases where dynamic graph compilation is necessary, however, the principles of Viterbi decoding can still be used. Recall that when pruning is used, the δ quantities are pushed forward to their successors in the graph. Essentially what is required for dynamic expansion is to associate enough information with each δ that its set of successor states can be computed on demand. This can be done in many ways, a good example being the strategy presented in [69].

5.2 Multipass Lattice Decoding

Under some circumstances, it is desirable to generate not just a single word hypothesis, but a set of hypotheses, all of which have some reasonable likelihood. There are a number of ways of doing this [69,90,65,71,98], and all result in a compact representation of a set of hypotheses as illustrated in Fig. 7. The states in a word lattice are annotated with time information, and the arcs with word labels. Additionally, the arcs may have the acoustic and language model scores associated with the word occurrence (note that with an n-gram LM, this implies that all paths of length $n-1$

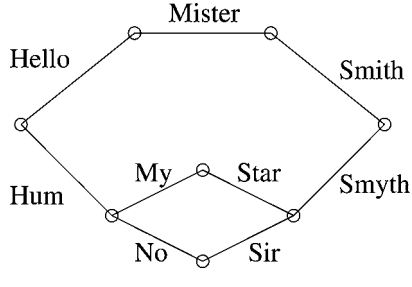


FIG. 7. A word lattice. Any path from the leftmost start state to the rightmost final state represents a possible word sequence.

leading into a state must be labeled with the same word sequence). We note also, that the posterior probability of a word occurrence in a lattice can be computed as the ratio of the sum likelihood of all the paths through the lattice that use the lattice link, to the sum likelihood of all paths entirely. These quantities can be computed with recursions analogous to the HMM $\alpha\beta$ recursions, e.g., as in [98].

Once generated, lattices can be used in a variety of ways. Generally, these involve recomputing the acoustic and language model scores in the lattice with more sophisticated models, and then finding the best path with respect to these updated scores. Some specific examples are:

- Lattices are generated with an acoustic model in which there is no cross-word acoustic context, and then rescored with a model using cross-word acoustic context, e.g., [58,46].
- Lattices are generated with a speaker-independent system, and then rescored using speaker-adapted acoustic models, e.g., [93].
- Lattices are generated with a bigram LM and then rescored with a trigram or 4-gram LM, e.g., [93,55].

The main potential advantage of using lattices is that the rescoring operations can be faster than decoding from scratch with sophisticated models. With efficient Viterbi implementations on static decoding graphs, however, it is not clear that this is the case [78].

5.3 Consensus Decoding

Recall that the decoding procedures that we have discussed so far have aimed at recovering the MAP word hypothesis:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} P(\mathbf{w}|\mathbf{a}) = \arg\max_{\mathbf{w}} \frac{P(\mathbf{w})P(\mathbf{a}|\mathbf{w})}{P(\mathbf{a})}.$$

Unfortunately, this is not identical to minimizing the WER metric by which speech recognizers are scored. The MAP hypothesis will asymptotically minimize *sentence* error rate, but not necessarily word error rate. Recent work [81,57] has proposed that the correct objective function is really the expected word-error rate under the posterior probability distribution. Denoting the reference or true word sequence by $\mathbf{r}$ and the string edit distance between $\mathbf{w}$ and $\mathbf{r}$ by $E(\mathbf{w}, \mathbf{r})$, the expected error is:

$$E_{P(\mathbf{r}|\mathbf{a})}\big[E(\mathbf{w}, \mathbf{r})\big] = \sum_{\mathbf{r}} P(\mathbf{r}|\mathbf{a})E(\mathbf{w}, \mathbf{r}).$$

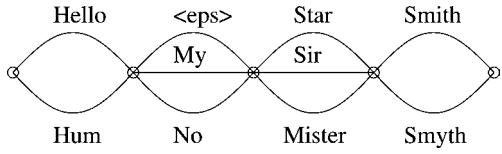


FIG. 8. A word lattice.

Thus, the objective becomes finding

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{\mathbf{r}} P(\mathbf{r}|\mathbf{a}) E(\mathbf{w}, \mathbf{r}).$$

There is no known dynamic programming procedure for finding this optimum when the potential word sequences are represented with a general lattice. Therefore, [57] proposes instead work with a segmental or sausage-like structure as illustrated in Fig. 8. To obtain this structure, the links in a lattice are clustered so that temporally overlapping and phonetically similar word occurrences are grouped together. Often, multiple occurrences of the same word (differing in time-alignment or linguistic history) end up together in the same bin, where their posterior probabilities are added together. Under the assumption of a sausage structure, the expected error can then be minimized simply by selecting the link with highest posterior probability in each bin [57]. This procedure has been widely adopted and generally provides a 5 to 10% relative improvement in large vocabulary recognition performance.

5.4 System Combination

In recent DARPA-sponsored speech recognition competitions, it has become common practice to improve the word error rate by combining the outputs of multiple systems. This technique was first developed in [21] where the outputs of multiple systems are aligned to one another, and a voting process is used to select the final output. This process bears a strong similarity to the consensus decoding technique, in that a segmental structure is imposed on the outputs, but differs in its use of multiple systems.

Although the problem of producing an optimal multiple alignment is NP complete [34,21] presents a practical algorithm for computing a reasonable approximation. The algorithm works by iteratively merging a sausage structure that represents the current multiple alignment with a linear word hypothesis. In this algorithm, the system outputs are ordered, and then sequentially merged into a sausage structure.

In a typical use [46], multiple systems are built differing in the front-end analysis, type of training (ML vs. MMI) and/or speaker adaptation techniques that are

used. The combination of 3 to 5 systems may produce on the order of 10% relative improvement over the best single system.

6. Adaptation

The goal of speaker adaptation is to modify the acoustic and language models in light of the data obtained from a specific speaker, so that the models are more closely tuned to the individual. This field has increased in importance since the early 1990s, has been intensively studied, and is still the focus of a significant amount of research. However, since no consensus has emerged on the use of language model adaptation, and many state-of-the-art systems do not use it, this section will focus solely on acoustic model adaptation. In this area, there are three main techniques:

- Maximum A Posteriori (MAP) adaptation, which is the simplest form of acoustic adaptation;
- Vocal Tract Length Normalization (VTLN), which warps the frequency scale to compensate for vocal tract differences;
- Maximum Likelihood Linear Regression, which adjusts the Gaussians and/or feature vectors so as to increase the data likelihood according to an initial transcription.

These methods will be discussed in the following sections.

6.1 MAP Adaptation

MAP adaptation is a Bayesian technique applicable when one has some reasonable expectation as to what appropriate parameter values should be. This prior $g(\theta)$ on the parameters θ is then combined with the likelihood function $f(\mathbf{x}|\theta)$ to obtain the MAP parameter estimates:

$$\theta^* = \arg\max_{\theta} g(\theta) f(\mathbf{x}|\theta).$$

The principled use of MAP estimation has been thoroughly investigated in [28], which presents the formulation that appears here.

The most convenient representation of the prior parameters for p-dimensional Gaussian mixture models is given by Dirichlet priors for the mixture weights $w_1, \ldots, w_K$, and normal-Wishart densities for the Gaussians (parameterized by means m_i and inverse covariance matrices r_i). These priors are expressed in terms of the following parameters:

- ν_k; a count $\nu_k > 0$,
- τ_k; a count $\tau_k > 0$,
- α_k; a count $\alpha_k > p - 1$,
- μ_k; a p dimensional vector,
- u_k; a $p \times p$ positive definite matrix.

Other necessary notation is:

- c_{kt}: the posterior probability of Gaussian k at time t,
- K: the number of Gaussians,
- n: the number of frames.

With this notation, the MAP estimate of the Gaussian mixture parameters are:

$$w'_k = \frac{\nu_k - 1 + \sum_{t=1}^{n} c_{kt}}{n - K + \sum_{k=1}^{K} \nu_k}, \quad m'_k = \frac{\tau_k \mu_k + \sum_{t=1}^{n} c_{kt} x_t}{\tau_k + \sum_{t=1}^{n} c_{kt}},$$

$$r_k'^{-1} = \frac{u_k + \tau_k (\mu_k - m'_k)(\mu_k = m'_k)^{\mathrm{T}}}{\alpha_k - p + \sum_{t=1}^{n} c_{kt}} + \frac{\sum_{t=1}^{n} c_{kt} (x_t - m'_k)(x_t = m'_k)^{\mathrm{T}}}{\alpha_k - p + \sum_{t=1}^{n} c_{kt}}.$$

Unfortunately, there are a large number of free parameters in the representation of the prior, making this formulation somewhat cumbersome in practice. [28] discusses setting these, but in practice it is often easier to work in terms of fictitious counts. Recall that in EM, the Gaussian parameters are estimated from first and second-order sufficient statistics accumulated over the data. One way of obtaining reasonable priors is simply to compute these over the entire training set without regard to phonetic state, and then to weight them according to the amount of emphasis that is desired for the prior. Similarly, statistics computed for one corpus can be downweighted and added to the statistics from another.

6.2 Vocal Tract Length Normalization

The method of VTLN is motivated by the fact that formants and spectral power distributions vary in a systematic way from speaker to speaker. In part, this can be viewed as a side-effect of a speech generation model in which the vocal tract can be viewed as a simple resonance tube, closed at one end. In this case the first resonant frequency is given by $1/L$, where L is the vocal tract length. While such a model is too crude to be of practical use, it does indicate a qualitative relationship between vocal tract length and formant frequencies. The idea of adjusting for this on a speaker-by-speaker basis is old, dating at least to the 1970s [85,7], but was revitalized by a CAIP workshop [44], and improved to a fairly standard form in [87]. The

basic idea is to warp the frequency scale so that the acoustic vectors of a speaker are made more similar to a canonical speaker-independent model. (This idea of "canonicalizing" the feature vectors will recur in another form in Section 6.3.2.) Figure 9 illustrates the form of one common warping function.

There are a very large number of variations on VTLN, and for illustration we choose the implementation presented in [87]. In this procedure, the FFT vector associated with each frame is warped according a warping function like that in Fig. 9. Ten possible warping scales are considered, ranging in the slope of the initial segment from 0.88 to 1.2. The key to this technique is to build a simple model of voiced speech, consisting of a single mixture of Gaussians trained on frames that are identified as being voiced. (This identification is made on the basis of a cepstral analysis described in [40].) To train the voicing model, each speaker is assigned an initial warp scale of 1, and then the following iterative procedure is used:

(1) Using the current warp scales for each speaker, train a GMM for the voiced frames.
(2) Assign to each speaker the warp scale that maximizes the likelihood of his or her warped features according to the current voicing model.
(3) Go to 1.

After several iterations, the outcome of this procedure is a voicing scale for each speaker, and a voicing model. Histograms of the voicing scales are generally bimodal, with one peak for men, and one for women. Training of the standard HMM parameters can then proceed as usual, using the warped or canonicalized features.

The decoding process in similar. For the data associated with a single speaker, the following procedure is used:

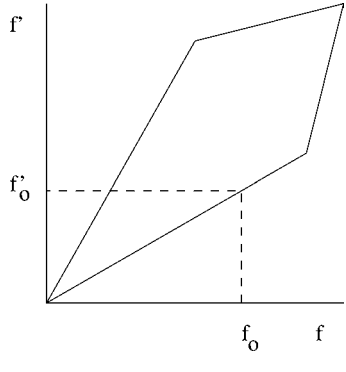


FIG. 9. Two VTLN warping functions. f_0 is mapped into f_0'.

(1) Select the warp scale that maximizes the likelihood of the warped features according to the voicing model.
(2) Warp the features and decode as usual.

The results reported in [87] indicate a 12% relative improvement in performance over unnormalized models, and improvements of this scale are typical [89,96].

As mentioned, a large number of VTLN variants have been explored. [37,61,89] choose warp scales by maximizing the data likelihood with respect to a full-blown HMM model, rather than a single GMM for voiced frames, and experiment with the size of this model. The precise nature of the warping has also been subject to scrutiny; [37] uses a piecewise linear warp with two discontinuities rather than one; [61] experiments with a power law warping function of the form

$$f' = \left(\frac{f}{f_N}\right)^{\beta} f_N$$

where f_N is the bandwidth and [96] experiments with bilinear warping functions of the form

$$f' = f + 2\arctan\left(\frac{(1-\alpha)\sin(f)}{1-(1-\alpha)\cos(f)}\right).$$

Generally, the findings are that piecewise linear models work as well as the more complex models, and that simple acoustic models can be used to estimate the warp factors.

The techniques described so far operate by finding a warp scale using the principles of maximum likelihood estimation. An interesting alternative presented in [20, 32] is based on normalizing formant positions. In [20], a warping function of the form

$$f' = k_s^{3f/8000}$$

is used, where k_s is the ratio of the speaker's third formant to the average frequency of the third formant. In [32], the speaker's first, second, and third formants are plotted against their average values, and the slope of the line fitting these points is used as the warping scale. These approaches, while nicely motivated, have the drawback that it is not easy to identify formant positions, and they have not been extensively adopted.

6.3 MLLR

A seminal paper [52] sparked intensive interest in the mid 1990s in techniques for adapting the means and/or variances of the Gaussians in an HMM model. Whereas VTLN can be thought of as a method for standardizing acoustics across speakers,

Maximum Likelihood Linear Regression was first developed as a mechanism for adapting the acoustic models to the peculiarities of individual speakers. This form of MLLR is known as "model-space" MLLR, and is discussed in the following section. It was soon realized [18,25], however, that one particular form of MLLR has an equivalent interpretation as on operation on the features, or "feature-space" MLLR. This technique is described in Section 6.3.2, and can be thought of as another canonicalizing operation.

6.3.1 Model Space MLLR

A well defined question first posed in [52] is, suppose the means of the Gaussians are transformed according to

$$\hat{\boldsymbol{\mu}} = A\boldsymbol{\mu} + \mathbf{b}.$$

Under the assumption of this form of transform, what matrix A and offset vector $\mathbf{b}$ will maximize the data probability given an initial transcription of the data? To solve this, one defines an extended mean vector

$$\boldsymbol{\xi} = [1\mu_1\mu_2 \ldots \mu_p]^{\mathrm{T}}$$

and a $p \times p+1$ matrix W. The likelihood assigned by a Gaussian g is then given by

$$N(\mathbf{x}; W\boldsymbol{\xi}_g, \Sigma_g).$$

In general, with a limited amount of training data, it may be advantageous to tie the transforms of many Gaussians, for example all those belonging to a single phone or phone-group such as vowels. If we define $\gamma_{g(t)}$ to be the posterior probability of Gaussian g having generated the observation o_t at time t, and $\mathcal{G}$ to be the set of Gaussians whose transforms are to be tied, then the matrix W is given by the following equation [52]

$$\sum_{t=1}^{t=N} \sum_{g \in \mathcal{G}} \gamma_g(t) \Sigma_g^{-1} \mathbf{o}(t) \boldsymbol{\xi}_g^{\mathrm{T}} = \sum_{t=1}^{t=N} \sum_{g \in \mathcal{G}} \gamma_g(t) \Sigma_g^{-1} W \boldsymbol{\xi}_g \boldsymbol{\xi}_g^{\mathrm{T}}.$$

Thus, estimating the transforms simply requires accumulating the sufficient statistics used in ML reestimation, and solving a simple matrix equation. Choosing the sets of Gaussians to tie can be done simply by clustering the Gaussians according to pre-defined phonetic criteria, or according to KL divergence [53]. Depending on the amount of adaptation data available, anywhere from 1 to several hundred transforms may be used.

A natural extension of mean-adaptation is to apply a linear transformation to the Gaussian variances as well [25,23]. The form of this transformation is given by

$$\hat{\boldsymbol{\mu}} = A\boldsymbol{\mu} + \mathbf{b} = W\boldsymbol{\xi}$$

and

$$\widehat{\Sigma} = H \Sigma H^{\mathrm{T}}$$

where W and H are the matrices to be estimated. A procedure for doing this is presented in [23].

6.3.2 Feature Space MLLR

Although it is a constrained version of the mean and variance transform described in the previous section, in some ways the most important form of MLLR applies the same transform to the means as to the variances:

$$\hat{\boldsymbol{\mu}} = A' \boldsymbol{\mu} - \mathbf{b}', \quad \widehat{\Sigma} = A' \Sigma A'^{\mathrm{T}}.$$

Under this constraint, straightforward estimation formulae can be derived, but more importantly, the transformation can be applied in to the feature vectors rather than the models, according to:

$$\hat{\mathbf{o}}(t) = A'^{-1} \mathbf{o}(t) + A'^{-1} \mathbf{b}' = A \mathbf{o}(t) + \mathbf{b}.$$

The likelihoods computed with this feature transformation differ from those computed with the model transform by $\log(|A|)$. When, as is often done, a single fMLLR transform is used, this can be ignored in Viterbi decoding and EM training. This has two important ramifications. Once the transforms have been estimated,

(1) Transformed features can be written out and the models can be retrained with the standard EM procedures (speaker-adaptive or SAT training) and
(2) MMI or other discriminative training can be performed with the transformed features.

Curiously, although multiple MLLR transforms are commonly used, the use of multiple fMLLR transforms has not yet been thoroughly explored. Due to the convenience of working with transformed or canonicalized features, feature space MLLR has become a common part of modern systems [78,93]. It is often used in conjunction with VTLN in the following speaker-adaptive or SAT training procedure:

(1) Train a speaker-independent (SI) system.
(2) Estimate VTLN warp scales using the frames that align to voiced phones with the SI system.
(3) Write out warped features for each speaker.
(4) Train a VTLN-adapted system.
(5) Estimate fMLLR transforms with the VTLN models.
(6) Write out fMLLR-VTLN features.
(7) Train ML and/or MMI systems from the canonical features.

7. Performance Levels

In order to illustrate the error rates attainable with today's technology—and the relative contribution of the techniques discussed in earlier sections—the following paragraphs describe the state-of-the-art as embodied by an advanced IBM system in 2002 [46]. This system was designed to work well across a wide variety of speakers and topics, and is tested on five separate datasets:

(1) Telephone conversations (Swb98).
(2) Meeting recordings (mtg).
(3) Two sets of call center recordings of customers discussing account information (cc1 and cc2).
(4) Voicemail recordings (vm).

In this system, the recognition steps are as follows:

P1 Speaker-independent decoding. The system uses mean-normalized MFCC features and an acoustic model with 4078 left context-dependent states and 171K mixture components.
P2 VTLN decoding. VTLN warp factors are estimated for each speaker using forced alignments of the data to the recognition hypotheses from P1, then recognition is performed with a VTLN system that uses mean-normalized PLP features and an acoustic model with 4440 left context-dependent states and 163K mixture components.
P3 Lattice generation. Initial word lattices are generated with a SAT system that uses mean-normalized PLP features and an acoustic model with 3688 word-internal context-dependent states and 151K mixture components. FMLLR transforms are computed with the recognition hypotheses from P2.
P4 Acoustic rescoring with large SAT models. The lattices from P3 are rescored with five different SAT acoustic models and pruned. The acoustic models are as follows:
  A An MMI trained PLP system with 10437 left context-dependent states and 623K mixture components. The maximum value of $c0$ is subtracted from each feature vector, and mean-normalization is performed for the other cepstral coefficients.
  B An MLE PLP system identical to the system of P4A, except for the use of MLE training of the acoustic model.
  C An MLE PLP system with 10450 left context-dependent states and 589K mixture components. This system uses mean normalization of all raw features including $c0$.
  D A SPAM MFCC system with 10133 left context-dependent states and 217K mixture components.

E An MLE MFCC system with 10441 left context-dependent states and 600K mixture components. This system uses max.-normalization of $c0$ and mean normalization of all other raw features.

The FMLLR transforms for each of the five acoustic models are computed from the one-best hypotheses in the lattices from P3.

P5 Acoustic model adaptation. Each of the five acoustic models are adapted with MLLR using one-best hypotheses from their respective lattices generated in P4.

P6 4-gram rescoring. Each of the five sets of lattices from P5 are rescored and pruned using a 4-gram language model.

P7 Confusion network combination. Each of the five sets of lattices from P6 are processed to generate confusion networks [57], then a final recognition hypothesis is generated by combining the confusion networks for each utterance.

The performance of the various recognition passes on the test set is summarized in Table I.

TABLE I
WORD ERROR RATES (%) FOR EACH TEST SET AT EACH PROCESSING STAGE AND THE OVERALL, AVERAGE ERROR RATE. FOR PASSES WHERE MULTIPLE SYSTEMS ARE USED (P4–6), THE BEST ERROR RATE FOR A TEST COMPONENT IS HIGHLIGHTED

Pass	swb98	mtg	cc1	cc2	vm	All
P1	42.5	62.2	67.8	47.6	35.4	51.1
P2	38.7	53.7	56.9	44.1	31.7	45.0
P3	36.0	44.6	46.6	40.1	28.0	39.1
P4A	**31.5**	**39.4**	41.7	38.2	26.7	35.5
P4B	32.3	40.0	**41.3**	39.0	26.7	35.9
P4C	32.5	40.2	42.1	39.9	27.0	36.3
P4D	31.7	40.3	42.6	**37.6**	**25.8**	35.6
P4E	33.0	40.5	43.4	38.8	26.9	36.5
P5A	30.9	**38.3**	**39.4**	36.9	26.1	34.3
P5B	31.5	38.5	**39.4**	37.0	26.5	34.6
P5C	31.6	38.7	41.0	39.4	26.8	35.5
P5D	**30.8**	39.0	41.1	**36.7**	**25.6**	34.6
P5E	32.1	38.9	41.8	36.8	26.4	35.2
P6A	**30.4**	**38.0**	**38.9**	36.5	25.7	33.9
P6B	31.0	38.3	**38.9**	36.4	25.8	34.1
P6C	31.2	38.4	40.1	38.9	26.3	35.0
P6D	**30.4**	38.6	40.8	36.3	**25.5**	34.3
P6E	31.5	38.5	41.6	**35.9**	25.7	34.6
P7	29.0	35.0	37.9	33.6	24.5	32.0

8. Conclusion

Over the past decade, incremental advances in HMM technology have advanced the state of the art to the point where commercial use is possible. These advances have occurred in all areas of speech recognition, and include

- LDA and HLDA analysis in feature extraction,
- discriminative training,
- VTLN, MLLR and FMLLR for speaker adaptation,
- the use of determinization and minimization in decoding graph compilation,
- consensus decoding,
- voting and system combination.

Collectively applied, these advances produce impressive results for many speakers under many conditions. However, under some conditions, such as when background noise is present or speech is transmitted over a low-quality cell phone or a speaker has an unusual accent, today's systems can fail. As the error-rates of Section 7 illustrate, this happens enough that the average error-rate for numerous tasks across a variety of conditions is around 30%—far from human levels. Thus, the most critical problem over the coming decade is develop truly robust techniques that reduce the error rate by another factor of five.

References

[1] Aho A.V., Sethi R., Ullman J.D., *Compilers: Principles, Techniques, and Tools*, Addison–Wesley, Reading, MA, 1986.
[2] Aubert X., "A brief overview of decoding techniques for large vocabulary continuous speech recognition", in: *Automatic Speech Recognition: Challenges for the New Millennium*, 2000.
[3] Axelrod S., Gopinath R., Olsen P., "Modeling with a subspace constraint on inverse covariance matrices", in: *ICSLP*, 2002.
[4] Bahl L.R., Brown P.F., de Souza P.V., Mercer R.L., "Maximum mutual information estimation of hidden Markov model parameters for speech recognition", in: *ICASSP*, 1986, pp. 49–52.
[5] Bahl L.R., et al., "Context dependent modeling of phones in continuous speech using decision trees", in: *Proceedings of DARPA Speech and Natural Language Processing Workshop*, 1991.
[6] Baker J., "The Dragon system—an overview", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **23** (1975) 24–29.
[7] Bamberg P., "Vocal tract normalization", Technical report, Verbex, 1981.

[8] Brown P.F., et al., "Class-based n-gram models of natural language", *Comput. Linguist.* **18** (1992).
[9] Chen S., Eide E., Gales M., Gopinath R., Olsen P., "Recent improvements in IBM's speech recognition system for automatic transcription of broadcast speech", in: *Proceedings of the DARPA Broadcast News Workshop*, 1999.
[10] Chen S.S., Gopalakrishnan P.S., "Clustering via the Bayesian information criterion with applications in speech recognition", in: *ICASSP*, 1995, pp. 645–648.
[11] Chen S., et al., "Speech recognition for DARPA communicator", in: *ICASSP*, 2001.
[12] Chen S.F., Goodman J., "An empirical study of smoothing techniques for language modeling", Technical Report TR-10-98, Harvard University, 1998.
[13] Chomsky N., *Aspects of the Theory of Syntax*, MIT Press, Cambridge, MA, 1965.
[14] CMU, *The CMU Pronouncing Dictionary*, 2003.
[15] Linguistic Data Consortium, *Callhome American English lexicon (pronlex)*, 2003.
[16] Davies K., et al., "The IBM conversational telephony system for financial applications", in: *Eurospeech*, 1999.
[17] Davis S., Mermelstein P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **28** (1980) 357–366.
[18] Digalakis V.V., Rtischev D., Neumeyer L.G., "Speaker adaptation using constrained estimation of Gaussian mixtures", *IEEE Transactions on Speech and Audio Processing* (1995) 357–366.
[19] Duda R.O., Hart P.B., *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
[20] Eide E., Gish H., "A parametric approach to vocal tract length normalization", in: *ICASSP*, 1996, pp. 346–348.
[21] Fiscus J.G., "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover)", in: *IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.
[22] Furui S., "Speaker independent isolated word recognition using dynamic features of speech spectrum", *IEEE Transactions on Acoustics Speech and Signal Processing* **34** (1986) 52–59.
[23] Gales M.J.F., "Maximum likelihood linear transformations for HMM-based speech recognition", Technical Report CUED-TR-291, Cambridge University, 1997.
[24] Gales M.J.F., "Maximum likelihood linear transformations for HMM-based speech recognition", *Computer Speech and Language* **12** (1998).
[25] Gales M.J.F., Woodland P.C., "Mean and variance adaptation within the MLLR framework", *Computer Speech and Language* **10** (1996) 249–264.
[26] Gao Y., Ramabhadran B., Chen J., Erdogan H., Picheny M., "Innovative approaches for large vocabulary name recognition", in: *ICASSP*, 2001.
[27] Gauvain J.-L., Lamel L., Adda G., "The LIMSI 1999 BN transcription system", in: *Proceedings 2000 Speech Transcription Workshop*, 2000, http://www.nist.gov/speech/publications/tw00/html/abstract.htm.

[28] Gauvain J.-L., Lee C.-H., "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains", *IEEE Transactions on Speech and Audio Processing* **2** (1994) 291–298.
[29] Gopalakrishnan P.S., Bahl L.R., Mercer R.L., "A tree-search strategy for large vocabulary continuous speech recognition", in: *ICASSP*, 1995.
[30] Gopalakrishnan P., Kanevsky D., Nadas A., Nahamoo D., "An inequality for rational functions with applications to some statistical estimation problems", *IEEE Transactions on Information Theory* **37** (1991) 107–113.
[31] Gopinath R., "Maximum likelihood modeling with Gaussian distributions for classification", in: *ICASSP*, 1998.
[32] Gouvea E.B., Stern R.M., "Speaker normalization through formant-based warping of the frequency scale", in: *Eurospeech*, 1997.
[33] Graff D., *The English Gigaword Corpus*, 2003.
[34] Gusfield D., *Algorithms on Strings, Trees and Sequences*, Cambridge Univ. Press, Cambridge, UK, 1997.
[35] Haeb-Umbach R., Ney H., "Linear discriminant analysis for improved large vocabulary continuous speech recognition", in: *ICASSP*, 1992.
[36] Hain T., Woodland P.C., Evermann G., Povey D., "The CU-HTK March 2000 HUB5E transcription system", in: *Proc. Speech Transcription Workshop*, 2000.
[37] Hain T., Woodland P.C., Niesler T.R., Whittaker E.W.D., "The 1998 HTK system for transcription of conversational telephone speech", in: *Eurospeech*, 1999.
[38] Hermansky H., "Perceptual linear predictive (PLP) analysis of speech", *J. Acoustical Society of America* **87** (1990) 1738–1752.
[39] Hopcroft J.E., Ullman J.D., *Introduction to Automata Theory, Languages and Computation*, Addison–Wesley, Reading, MA, 1979.
[40] Hunt M.J., "A robust method of detecting the presence of voiced speech", in: *ICASSP*, 1995.
[41] Jan E., Maison B., Mangu L., Zweig G., "Automatic construction of unique signatures and confusable sets for natural language directory assistance applications", in: *Eurospeech*, 2003.
[42] Jeffreys H., *Theory of Probability*, Clarendon, Oxford, 1948.
[43] Jelinek F., "Continuous speech recognition by statistical methods", *Proceedings of the IEEE* **64** (1976) 532–556.
[44] Kamm T., Andreou A., Cohen J., "Vocal tract normalization in speech recognition: Compensating for systematic speaker variability", in: *Proceedings of the 15th Annual Speech Recognition Symposium, Baltimore, MD*, 1995, pp. 175–178.
[45] Katz S.M., "Estimation of probabilities from sparse data for the language model component of a speech recognizer", *IEEE Transactions of Acoustics, Speech and Signal Processing* **35** (1987) 400–401.
[46] Kingsbury B., Mangu L., Saon G., Zweig G., Axelrod S., Visweswariah K., Picheny M., "Towards domain independent conversational speech recognition", in: *Eurospeech*, 2003.
[47] Kneser N., Ney H., "Improved backing-off for m-gram language modeling", in: *ICASSP*, 1995.

[48] Kuhn R., "Speech recognition and the frequency of recently used words: A modified Markov model for natural language", in: *12th International Conference on Computational Linguistics, Budapest*, 1988, pp. 348–350.
[49] Kuhn R., De Mori R., "A cache based natural language model for speech recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12** (1990) 570–583.
[50] Kumar N., Andreou A.G., "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition", *Speech Communication* (1998) 283–297.
[51] Leggetter C., Woodland P.C., "Flexible speaker adaptation using maximum likelihood linear regression", in: *Eurospeech*, 1995.
[52] Leggetter C., Woodland P.C., "Speaker adaptation of continuous density HMMs using multivariate linear regression", in: *ICSLP*, 1994.
[53] Leggetter C.J., Woodland P.C., "Flexible speaker adaptation using maximum likelihood linear regression", in: *Eurospeech*, 1995.
[54] Levinson S.E., Rabiner L.R., Sondhi M.M., "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition", *The Bell System Technical Journal* **62** (1983) 1035–1074.
[55] Ljolje A., et al., "The AT&T 2000 LVSCR system", in: *Proceedings 2000 Speech Transcription Workshop*, 2000, http://www.nist.gov/speech/publications/tw00/html/abstract.htm.
[56] Makhoul J., "Linear prediction: A tutorial review", *Proceedings of the IEEE* **63** (1975) 561–580.
[57] Mangu L., Brill E., Stolcke A., "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks", *Computer Speech and Language* **14** (2000) 373–400.
[58] Matsoukas S., et al., "Speech to text research at BBN", in: *Proceedings of January 2003 EARS Midyear Meeting*, 2003.
[59] Mohri M., "Finite-state transducers in language and speech processing", *Comput. Linguist.* **23** (1997).
[60] Mohri M., Riley M., Hindle D., Ljolje A., Pereira F., "Full expansion of context-dependent networks in large vocabulary speech recognition", in: *ICASSP*, 1998.
[61] Molau S., Kanthak S., Ney H., "Efficient vocal tract normalization in automatic speech recognition", in: *ESSV*, 2000, pp. 209–216.
[62] Nadas A., "A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus unconditional maximum likelihood", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **31** (1983).
[63] Nadas A., Nahamoo D., Picheny M., "On a model-robust training method for speech recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing* **36** (1988).
[64] Naur P., "Revised report on the algorithmic language Algol 60", *Communications of the Association for Computing Machinery* **6** (1963) 1–17.
[65] Neukirchen C., Klakow D., Aubert X., "Generation and expansion of word graphs using long span context information", in: *ICASSP*, 2001.
[66] Ney H., Essen U., Kneser R., "On structuring probabilistic dependences in stochastic language modelling", *Computer Speech and Language* (1994) 1–38.

[67] Niesler T.R., Whittaker E.W.D., Woodland P.C., "Comparison of part-of-speech and automatically derived category-based language models for speech recognition", in: *ICASSP*, 1998.

[68] Normandin Y., Regis C., De Mori R., "High-performance connected digit recognition using maximum mutual information", *IEEE Transactions on Speech and Audio Processing* **2** (1994) 299–311.

[69] Odell J.J., "The use of context in large vocabulary speech recognition", Cambridge University dissertation, 1995.

[70] Olsen P., Gopinath R., "Extended MLLT for Gaussian mixture models", *IEEE Transactions on Speech and Audio Processing* (2001).

[71] Ortmanns S., Ney H., "A word graph algorithm for large vocabulary continuous speech recognition", *Computer Speech and Language* (1997) 43–72.

[72] Pellom B., Ward W., Hansen J., Hacioglu K., Zhang J., Yu X., Pradhan S., "University of Colorado dialog systems for travel and navigation", in: *Human Language Technologies*, 2001.

[73] Rabiner L.R., Juang B.-H., "An introduction to hidden Markov models", *IEEE ASSP Magazine* (1986) 4–16.

[74] Rabiner L.R., Juang B.-H., *Fundamentals of Speech Recognition*, Prentice Hall, New York, 1993.

[75] Rosenfeld R., "A maximum entropy approach to adaptive statistical language modeling", *Computer Speech and Language* **10** (1996) 187–228.

[76] Sankar A., Gadde V.R.R., Stolcke A., Weng F., "Improved modeling and efficiency for automatic transcription of broadcast news", *Speech Communication* **37** (2002) 133–158.

[77] Saon G., Padmanabhan M., Gopinath R., Chen S., "Maximum likelihood discriminant feature spaces", in: *ICASSP*, 2000.

[78] Saon G., Zweig G., Kingsbury B., Mangu L., Chaudhari U., "An architecture for rapid decoding of large vocabulary conversational speech", in: *Eurospeech*, 2003.

[79] Schroeder M.R., "Recognition of complex acoustic signals", in: Bullock T.H. (Ed.), *Life Sciences Research Report 5*, Abakon Verlag, 1977.

[80] Seymore K., Rosenfeld R., "Scalable backoff language models", in: *ICSLP*, 1996.

[81] Stolcke A., Konig Y., Weintraub M., "Explicit word error minimization using n-best list rescoring", in: *Eurospeech*, 1997.

[82] Stolcke A., "Entropy-based pruning of backoff language models", in: *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[83] Stolcke A., "Srilm—an extensible language modeling toolkit", in: *ICSLP*, 2002.

[84] Suontausta J., Hakkinen J., Olli V., "Fast decoding in large vocabulary name dialing", in: *ICASSP*, 2000, pp. 1535–1538.

[85] Waitika H., "Normalization of vowels by vocal-tract length and its application to vowel identification", *IEEE Transactions on Audio Speech and Signal Processing* (1977) 183–192.

[86] Wegmann S., Zhan P., Carp I., Newman M., Yamron J., Gillick L., "Dragon systems' 1998 broadcast news transcription system", in: *Proceedings of the DARPA Broadcast News Workshop*, NIST, 1999.

[87] Wegmann S., McAllaster D., Orloff J., Peskin B., "Speaker normalization on conversational telephone speech", in: *ICASSP*, 1996.

[88] Welling L., Haberland N., Ney H., "Acoustic front-end optimization for large vocabulary speech recognition", in: *Eurospeech*, 1997.

[89] Welling R., Haeb-Umbach R., Aubert X., Haberland N., "A study on speaker normalization using vocal tract normalization and speaker adaptive training", in: *ICASSP*, 1998, pp. 797–800.

[90] Weng F., Stolcke A., Sankar A., "Efficient lattice representation and generation", in: *ICSLP*, 1998.

[91] Whittaker E.W.D., Woodland P.C., "Efficient class-based language modelling for very large vocabularies", in: *ICASSP*, 2001.

[92] Woodland P.C., Povey D., "Large scale discriminative training for speech recognition", in: *Automatic Speech Recognition: Challenges for the New Millennium*, 2000.

[93] Woodland P., et al., "The CU-HTK April 2002 switchboard system", in: *EARS Rich Transcription Workshop*, 2002.

[94] Young S., Odell J., Ollason D., Valtchev V., Woodland P., *The HTK Book*, 2.1 edition, Entropic Cambridge Research Laboratory, 1997.

[95] Young S.J., Odell J.J., Woodland P.C., "Tree-based tying for high accuracy acoustic modelling", in: *ARPA Workshop on Human Language Technology*, 1994.

[96] Zhan P., Waibel A., "Vocal tract length normalization for large vocabulary continuous speech recognition", Technical Report CMU-CS-97-148, School of Computer Science, Carnegie Mellon University, 1997.

[97] Zue V., et al., "A telephone-based conversational interface for weather information", 2000.

[98] Zweig G., Padmanabhan M., "Exact alpha–beta computation in logarithmic space with application to map word graph construction", in: *ICSLP*, 2000.

[99] Zweig G., Saon G., Yvon F., "Arc minimization in finite state decoding graphs with cross-word acoustic context", in: *ICSLP*, 2002.

[100] Zwicker E., "Subdivision of the audible frequency range into critical bands", *J. Acoustical Society of America* **33** (1961) 248.

[101] Zwicker E., "Masking and physiological excitation as consequences of ear's frequency analysis", in: Plomp R., Smoorenburg G.F. (Eds.), *Frequency Analysis and Periodicity Detection in Hearing*, 1970.

This page intentionally left blank

Author Index

Numbers in *italics* indicate the pages on which complete references are given.

C

M

Subject Index

S

This page intentionally left blank

Contents of Volumes in This Series

Volume 40

Volume 41

Volume 42

Volume 43

Volume 44

Volume 45

Volume 46

Volume 47

Volume 48

Volume 49

Volume 50

Volume 51

Volume 52

Volume 53

Volume 56

Volume 57

Volume 58

Volume 59

This page intentionally left blank