COMMUNICATIONS ENGINEERING SERIES

Bulletproof Wireless Security

GSM, UMTS, 802.11, and Ad Hoc Security

Praphul Chandra

Newnes

Bulletproof Wireless Security

Bulletproof Wireless Security

GSM, UMTS, 802.11 and Ad Hoc Security

By

Praphul Chandra

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Newnes is an imprint of Elsevier

Newnes is an imprint of Elsevier
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
Linacre House, Jordan Hill, Oxford OX2 8DP, UK

Library of Congress Cataloging-in-Publication Data

(Application submitted.)

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

ISBN: 0-7506-7746-5

05 06 07 08 09 10 10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

This book is dedicated—

To my parents,
whose love, support and example
have helped me reach my goals;

and

To my wife, Shilpy,
whose cheerful patience and
constant encouragement
made this book possible.

Contents

Preface

"... but where does the voice go?"

A man has been arrested in New York for attempting to extort funds from ignorant and superstitious people by exhibiting a device which he says will convey the human voice any distance over metallic wires so that it will be heard by the listener at the other end. He calls this instrument a telephone. Well-informed people know that it is impossible to transmit the human voice over wires."

—News item in a New York newspaper, 1868.

I remember a day not so long ago, when I was showing my mother how to use a cell phone. She asked me how it worked and I started describing the base-stations, switches and the cellular architecture. After I was done, she asked me "... but where does the voice go?"—Where indeed?

Engineers sometimes tend to forget that the concept of *wireless* seems magical to most people. Being visual creatures, we can accept the fact that in a wired network, our voice (or data) travels "on" the wire but seeing a wireless network is almost magical—your voice (or data) disappears into your handset and reappears out of the handset of another person at the other side of the globe.

Pause to think about it. If you told someone in the nineteenth century that you could do this, you would have probably been credited with supernatural powers. We really are doing magic. I had always been fascinated by wireless. Then during my college years, I took a course in cryptography and was intrigued by how secure communication could be achieved over an unsecure channel. It was only natural then that the field of wireless security attracted me towards it and resulted in this book.

I have always felt that for a complete understanding of any field, it helps to know how the field developed. We therefore start by looking at a brief history of wireless and a brief history of cryptography. Those with a purely technical inclination may, therefore, skip this chapter but I think it makes for good light reading. I hope you enjoy this chapter and the rest of this book.

A Brief History of Wireless

There is no doubt that the day will come, maybe when you and I are forgotten, when copper wires, gutta-percha coverings, and iron sheathings will be relegated to the Museum of Antiquities. Then, when a person wants to telegraph to a friend, he knows not where, he will call an electromagnetic voice, which will be heard loud by him who has the electromagnetic ear, but will be silent to everyone else. He will call "Where are you?" and the reply will come, "I am at the bottom of the coal-mine" or "Crossing the Andes" or "In the middle of the Pacific"; or perhaps no reply will come at all, and he may then conclude that his friend is dead.

—Professor W.E. Ayrton (member of the Institution of Electrical Engineers) said this at a lecture at the Imperial Institute...in 1897.

Arguably, wireless communication between humans is as old as the human civilization itself, for as soon as the first humans started communicating with each other using their vocal cords, we had achieved wireless communication. However, the term *wireless communication* is usually used to refer to wireless communication beyond the "line of sound."

The foundations of wireless communication were laid by Michael Faraday's work on electromagnetism, which established that electric and magnetic effects result from "lines of force" that surround conductors and magnets. Based on Faraday's work, James Maxwell derived mathematical equations that represented the "lines of force" Faraday had explained. Maxwell published his work in a paper in 1855. Later, in 1861, Maxwell further developed his work showing that if an electric charge was applied to a (hypothetical) elastic fluid, it would result in the generation of waves that would travel through the medium. In effect, Maxwell predicted the existence of electromagnetic waves. Friedrich Kohlrausch and Wilhem Weber furthered Maxwell's work by calculating that these waves would travel at the speed of light.

Up until 1888, the field of electromagnetism was that of pure theory. In that year, Heinrich Hertz discovered radio waves which are an example of electromagnetic radiation. Hertz did this by devising a transmitting oscillator and a "receiver." The "receiver" was basically a metal loop with a gap on one side. When this loop was placed within the transmitter's electromagnetic field, sparks were produced across the gap in the loop. This proved that electromagnetic waves could be sent out into space and remotely detected. In effect, Hertz showed that the elastic fluid that Maxwell had hypothesized could be the ether. The discovery of radio waves confirmed the ideas of Maxwell and other scientists who had worked on electromagnetism and sparked a greater interest in the field.

When Guglielmo Marconi learnt about Hertz's work, he realized that if the radio waves could be transmitted over large distances, wireless telegraphy could be developed. Marconi started experimenting with this idea and by 1894, he managed to receive radio signals at a distance of over a mile. Marconi tried to develop his work further by taking the help of the Italian government. However, the Italian government was not interested. So, Marconi approached the British government. He was granted a patent for wireless telegraphy in 1897 and the world's first radio factory was setup at Chelmsford in 1898. Soon, radios started to be used commercially. The world of wireless telegraphy got another big boost in 1901 when Marconi and his associates were able to receive a signal across the Atlantic successfully. Recognizing his contribution to the field of wireless communication, Marconi was awarded the Nobel Prize in 1909.

In 1914, physicists were able to use radio transmission to carry voice and by the 1920s, wireless mobile receivers were being installed in police cars in Detroit. Commercially, wireless deployment reached its first landmark in 1983 with the deployment of the Advanced Mobile Phone System (AMPS) in the United States. AMPS was an example of the first generation wireless networks that were deployed across the world. Although a major success story, the 1G (first generation) wireless networks soon outgrew the capacity needed to serve the exploding growth in the number of wireless subscribers. This motivated the development and deployment of the 2G (second generation) wireless networks like GSM in the late 1990s. Today, 2G is the dominant mobile technology. The deployment of 3G is expected to begin soon[1], but on another note, the exploding growth in Wireless Local Area Networks (WLANs) is changing the field of wireless communication in unforeseen ways. As of the writing of this book, pundits are trying to envision how 3G, IP, PSTN and WLANs will come together to provide the ultimate communication dream—staying connected: anytime, anywhere.

[1] Some may argue that 3G may never happen and service providers may go straight from 2.5G to 4G. See Chapter 4 for more details.

A Brief History of Security

"Well, I never heard it before," said the Mock Turtle, "but it sounds uncommon nonsense."

—Lewis Carroll, *Alice in Wonderland.*

Secret communication achieved by hiding the existence of a message is known as steganography. The word is derived from the Greek word "steganos," meaning covered and "graphin" meaning to write.

The first written account of steganography comes from Herodotus, who chronicled the story of Histaiaeus. Histaiaeus wanted to encourage Aristagoras of Miletus to revolt against the Persian king. To convey his instructions securely, Histaiaeus shaved the head of his messenger, wrote the message on his scalp and then waited for the hair to regrow. From that humble beginning, steganography evolved to the microdot in World War II. The microdot was a technique wherein the German agents in Latin America would photographically shrink a page of text down to a dot less than 1 mm in diameter, and then hide this microdot on top of a full stop in an apparently innocuous letter. The first microdot to be spotted by the Federal Bureau of Investigation (FBI) (USA) was in 1941. The coming of the digital age further changed the face of steganography. Modern techniques involve hiding the content of a message in a picture by modifying the lower nibble of a pixel.

Whereas steganography deals with hiding the message, the other branch of secret communication, cryptography, deals with hiding the information content of the message. Cryptography consists of two basic operations—transposition and substitution. Transposition involves rearranging the "letters" in the message and substitution involves mapping the "letters" in a message according to a predetermined mapping. In cryptographic lingo, the new message obtained by transforming the original message using cryptography is known as the ciphertext, whereas the original message is known as the plaintext.

The transformation of the plaintext to the ciphertext is achieved using a cipher. Each distinct cipher can be described in terms of the algorithm and the key. As an example, consider the Caesar cipher, one of the earliest military ciphers used by Julius Caesar. This cipher works by replacing each letter in the message with a letter which is three places down the alphabet. In this case, the algorithm part of the cipher is the act of substitution and the key is "three forward." More generically, the mono-alphabetic substitution cipher is the generic name given to any substitution cipher in which each letter in the plaintext is replaced by exactly one letter or symbol in the ciphertext.

The first documented use of mono-alphabetic substitution cryptography appears in the Kama Sutra, a text written in the fourth century B.C. by the Indian scholar Vatsyayna. Vatsyayna explains a technique of secret writing that works by pairing the letters of the alphabet at random and then substituting each letter in the message with its partner. With the passage of time, multiple variations of the mono-alphabetic cipher continued to be developed independently around the world.

The next biggest invention in the world of "secret writing" came with the invention of cryptanalysis, or the science of destroying ciphers. Cryptanalysis consists of obtaining the plaintext message from the ciphertext without the knowledge of the key. The invention of cryptanalysis can be traced back to the ninth century. In 815, the Caliph al-Mamun established the Bait al-Hikmah ("House of Wisdom") in Baghdad and assigned Arabic theologians to scrutinize the revelations of Muhammad the Prophet and establish the chronology of the revelations.

The Arabic theologians did this by counting the frequencies of words contained in each revelation. The theory was that certain words had evolved relatively recently; and hence, if a revelation contained a high number of these new words, this would indicate that it came later in the chronology. Significantly, the scholars did not stop their scrutiny at the level of words. They also analyzed individual letters and discovered that some letters are more likely to occur in a given text than others.

Although it is not known who first realized that the variation in the frequencies of letters could be exploited in order to break ciphers, the earliest known description of this technique is by the ninth-century scientist Abu Yusuf Yaqub ibn Is-haq ibn as-Sabbah ibn omran ibn Ismail al-Kindi. The cryptanalysts had triumphed over the cryptographers and thus began the "war" between cryptographers trying to build unbreakable ciphers and cryptanalysts trying to break these ciphers. As we shall see, this war continues to this day and provides the impetus for the evolution of cryptography.

The onus was now on the cryptographers to come up with a new stronger cipher. The roots of this new stronger type of cipher can be traced back to an essay written sometime in the 1460s by the Florentine polymath Leon Battista Alberti. Alberti proposed mapping each plaintext letter to two or more ciphertext letters and switching between them during the encipherment. Although Alberti had hit upon the most significant breakthrough in cryptography for over a thousand years, he failed to develop this concept into a complete cipher.

Alberti made one other significant contribution to the field of cryptography—he invented the first cryptographic machine—the cipher disc. The cipher disc is the earliest known cryptographic machine and it consists of two concentric copper discs, one slightly larger than the other, with the alphabet inscribed along the circumference of both the discs. The smaller disc is placed on top of the larger disc and connected at the center using a needle which acts as an axis. Since the two discs could be independently rotated, the two alphabets can have different relative positions and can therefore easily be used to encrypt messages using the mono-alphabetic cipher. In fact, the disk can also be used in more complicated ways. Alberti suggested changing the setting of the disk while encrypting the message to use the poly-alphabetic cipher that he had just invented.

Alberti's initial idea regarding the poly-alphabetic cipher was further developed by Johannes Trithemius and Giovanni Porta over the years. However, the development of this idea to a complete cipher was left to Blaise de Vigenere. Although Alberti, Trithemius and Porta all made vital contributions to this new poly-alphabetic cipher, the cipher is known as the Vigenere cipher in honor of the man who developed it into its final form. The strength of the Vigenere cipher lies in it's using not one but 26 distinct cipher alphabets to encrypt a message. The great advantage of Vigenere cipher is that it is impregnable to the frequency analysis which the cryptanalysts had used to break the mono-alphabetic cipher. The fact that a letter which appears several times in the ciphertext can represent a different plaintext letter on each occasion generates tremendous ambiguity for the cryptanalyst. Besides being invulnerable to frequency analysis, the Vigenere cipher also has an enormous number of keys, making it difficult to try all possible keys. The creation of the poly-alphabetic Vigenere cipher meant that the cryptographers were now in control.

The Vigenere cipher remained unbreakable until the mid-nineteenth century. This is when Charles Babbage came along. Babbage is best known for developing the blueprint of the modern computer—the Difference Engine. However, Babbage also made the greatest breakthrough in cryptanalysis since the Arab scholars in the ninth century—he broke the Vigenere cipher. Babbage never publicized this discovery. His discovery came to light only in the twentieth century when scholars examined Babbage's extensive notes. Meanwhile, in 1863 Friedrich Wilhelm Kasiki also broke the Vigenere cipher independently and published his discovery. The breaking of Vigenere cipher put the cryptanalysts back on the top.

Since the Vigenere cipher was broken, the cryptographers had been trying to come up with a better, more secure type of cipher. The need for such a cipher grew in the late nineteenth century with the invention of the telegraph and the radio. The use of the telegraph took the speed of communications to new heights. However, for businessmen and the military to exploit the immediacy of the telegraph required the use of an unbreakable cipher, since messages sent using the telegraph ended up being handled by a whole group of people (telegraph operators, and so forth).

The demand for a secure cipher was further fueled by the invention of radio by the Italian physicist Guglielmo Marconi. Wireless communication was desirable for many reasons, especially by the military. Primary among them were that communication could be achieved with minimal infrastructure support and that communication could be achieved even if the communicating parties were constantly moving. These advantages were inherent due to the all-pervasive nature of radio. However, the all-pervasive property of the radio was also its greatest weakness, since this meant that the messages sent from the transmitter to the receiver in the battlefield were also accessible to the enemy nearby. Consequently, an unbreakable cipher became an absolute necessity.

However, the discovery of the next great cipher was not to come until 1918, and the field of cryptography did not see any major advances during World War I (1914–1918). The field of cryptanalysis though, was another story. During the war, the French listening ports learnt to recognize a radio operator's fist (his pauses, his speed of transmission and his relative lengths of dots and dashes). The French also established six direction finding stations which were able to detect the direction from which a radio message was coming. Since each enemy battalion usually had an assigned radio operator and since battalions were mobile, the above two pieces of information could be combined to track the movement of enemy battalions. This was probably the birth of traffic analysis as a form of cryptanalysis, and during the war this became an especially valuable tool when a new cipher was introduced by the enemy. In fact, the French also recognized that wireless communication was more unsecure than wired communication due to the ease of message collection and exploited this fact by destroying communication landlines as they retreated. This forced the advancing Germans to use radio communication; thus making message collection easier for the French.

In short, World War I was dominated by the cryptanalysts. New ciphers were introduced but all of them were broken one by one. Then, in 1918, Major Joseph Mauborgne, head of cryptographic research for the US Army, introduced the concept of a random key. The idea was inspired by the fact that the fundamental weakness of the Vigenere cipher that was exploited by Babbage and Kasiki to break it was the cyclical nature of the cipher when used with a short key. Since the key was limited in length, every *n*th letter of the plaintext was encrypted according to the same ciphertext alphabet. Mauborgne advocated employing message-length random keys as part of a

Vigenere cipher to give an unprecedented level of security. This cipher was known as the one-time pad cipher since it required the generation of large “pads” of random keys.

The security of the one-time pad cipher lay wholly in the randomness of the key. The key injects randomness into the ciphertext and if the ciphertext is truly random, there is no structure for the cryptanalyst to exploit. In fact, to date the one-time pad cipher is the only cipher which can be mathematically proven to be absolutely secure. At first thought, this may lead one to believe that the cryptographers had once and for all won the war against the cryptanalysts. If this were true, this book never would have been written. Perfectly secure as the one-time pad is, it suffers from two great operating difficulties—key generation and key distribution.

Generating truly random keys is not as easy as it might initially sound. As Voltaire put it, “Anybody who tries to generate random numbers by deterministic means is of course living in a state of sin.” Over the years, cryptographers have realized that the best random keys are those created by harnessing natural physical processes like radioactivity. The bottom line is that it requires a great deal of time, effort and money to generate truly random keys. Difficult as key generation was, there was another major problem with the one-time pad cipher—the distribution of these large pads of keys. To be fair, key distribution had always been a problem in the world of cryptography—and a neglected one at that. The one-time pad cipher just brought the problem into the limelight by making it a lot more difficult to solve (due to the sheer volume of the pads).

Even though the cryptographers had created the perfect cipher in 1918, it was of little use due to its huge operating cost. However, there was another development in 1918 that changed the field of cryptography. This was the development of the Enigma by the German inventor, Arthur Scherbius. The Enigma was a cryptographic machine which could be used for encrypting and decrypting messages. It was an electrical version of Alberti’s cipher disc but was much more powerful. A user could simply type in the plaintext alphabet (as in the keyboard of a typewriter) and obtain the corresponding ciphertext. The cryptographic core of the Enigma was the scrambling unit, which consisted of a set of scrambler discs (also known as rotors). The first disc automatically rotated by one-sixth of a revolution each time a letter was encrypted. The second disk rotated each time the first disc had completed a revolution, and so on.

It helps to think of the Enigma in terms of Alberti’s cipher disc. Enigma had combined the scramblers to implement a poly-alphabetic cipher which continually switched between different cipher alphabets. Consider what would happen if the inner disc of Alberti’s cipher disc was rotated after the encryption of each letter: we would have a poly-alphabetic cipher with a self-generating key. In fact, the length of the “key” would be as long as the message itself. So, was this the implementation of a one-time pad cipher? Well, not quite. Remember that the one-time pad cipher requires the key to be random. In the Enigma the generation of the "key" was a factor of the initial settings of the scramblers and the plaintext itself. Even though this made the discovery of the key very tough, the key was not really random—it was just mechanically (and therefore, mathematically) convoluted. Note that even the Enigma was faced with the problem of key distribution. In case of the Enigma, even though the key was generated during encryption, the initial settings of the scrambler needed to be known to the sender(s) and the receiver(s) before secure communication would begin. However, the initial settings of the Enigma could be changed on a periodic basis—daily, weekly, and so forth. This made the amount of data that needed to be securely distributed much less than that required for a one-time pad cipher.

The invention of the Enigma was truly a great one, and one which changed the face of cryptography forever. The scrambler orientations, arrangements and the plug-board settings together offered a possible of 10,000,000,000,000,000 variations of the initial arrangements from which the cryptanalyst would have to search to break the cipher. The cryptographers were back on top in their battle with the cryptanalysts. To be fair, Scherbius was not the only one who had hit upon the idea of rotating scramblers. Alexander Koch in the Netherlands and Arvind Damm in Sweden had independently and almost simultaneously hit upon this idea. However, none of them could market the machine well enough to make it a commercial success. It was only in 1927, when the Germans realized that the Achilles heel of their World War I campaign was the breaking of their cipher, that the German government selected the Enigma for use by the military. In fact, the Enigma was to play a crucial role in World War II.

Enigma was at the heart of Hitler's blitzkieg (literally—*lightning war*) strategy, whose ethos was "speed of attack through the speed of communication." When the Americans and the French began to encounter messages encrypted with the Enigma, they were completely baffled and quickly gave up. This was probably due to the fact that in the wake of the victory in World War I, the Allies were in a dominant position and feared no one, least of all Germany. There was, therefore, no great motivation for the Allies' cryptanalysts. There was one country, though, that could not afford to relax—Poland. After World War I, Poland had reestablished itself as an independent state. However, to the east of Poland lay Russia, a nation ambitious to spread its communism, and to the west lay Germany, desperate to regain territory ceded to Poland after the war. The Polish cryptanalysts therefore had plenty of motivation to attack the Enigma. Adversity, it seems, is one of the foundations of successful code breaking.

The first step in the cracking of the Enigma came on November 8, 1931 when Hans-Thilo Schmidt, a German military employee, handed over German military documents containing instructions for using the military version of the Enigma machine, to the French in exchange for 10,000 marks. These documents contained enough information to allow the Allies to create an exact replica of the Enigma machine. However, the French cryptanalysts did not actually build a replica, since they were convinced that the next stage of breaking the cipher—that is, finding the key required to decipher a message—was impossible. This underscores one of the basic presumptions of cryptography—the strength of a good cipher depends not on keeping the algorithm (machine) secret but on keeping the key (initial setting of the machine) secret.

Fortunately for the Allies, the French cryptanalysts handed over the Schmidt documents to the Polish cryptanalysts thanks to an existing military co-operation agreement between the two countries. This allowed the Polish cryptanalysts to attack the Enigma even though the French cryptanalysts had given up hope. In attacking the Enigma, the Polish bureau of cryptography made a breakthrough in the field of cryptography. For centuries, it had been assumed that the best cryptanalysts were experts in the structure of language. However, the Polish theorized that since Enigma was a mechanical cipher, cryptanalysts with a more scientific mind might be better equipped to attack the Enigma. Working off this theory, the Polish bureau assigned three mathematicians to work on the Enigma cipher—one of them was Marian Rejewski.

Working alone on the Enigma, Rejewski's attack was based on the theory that repetition is the enemy of security, since repetition leads to patterns which can be linked and therefore exploited. The Schmidt documents showed that the most obvious repetition in the Enigma cipher was the message key. To discourage attacks on the Enigma, the German cryptographers had decided that instead of using the day key (initial scrambler settings) to encrypt all messages of the day, the day key was instead used just to encrypt the message key. The message key was used to encrypt the message itself and the message key itself was transmitted encrypted with the day key. This process made the system more secure by avoiding the repeated use of a single day key to encrypt hundreds of messages. However, the fact that the message key was encrypted and transmitted twice at the beginning of every message to avoid mistakes caused by radio interference made the process susceptible to Rejewski's repetition theory.

Even though an explanation of Rejewski's breakthrough in cracking the Enigma is beyond the scope of this book, it is imperative to mention here that Rejewski's attack on Enigma is one of the truly great accomplishments of cryptanalysis. Not only did Rejewski break one of the greatest ciphers of the time, he also achieved another first in the cryptographic world by mechanizing the cryptanalysis. The machines that were used to automate the process of finding the day key were called *bombes*, a name attributed to the ticking noise they made while working. For the Allies, the cracking of the Enigma was a major breakthrough since it counteracted the ethos of Blitzkrieg, which was "speed of attack through speed of communications."

However, in December of 1938 when the Germans added two more scramblers to the Enigma, Rejewski's bombes were rendered useless and the Polish did not have the resources to tackle the new, even more complicated, cipher. Cracking the new Enigma was now left to Britain and France, which used Rejewski's breakthrough as a starting step in this process. Bletchley Park in Buckinghamshire was selected as the center of Allied code-breaking with Enigma being one of the major challenges. Learning from the Polish experience, the team at Bletchley Park consisted of mathematicians and scientists, besides linguists and classicists. Working with the new Enigma cipher, over time, the Bletchley team invented their own short cuts for finding the Enigma message-keys.

There were many significant breakthroughs at Bletchley Park but the one that most deserves mention is that of Alan Turing. Turing is most known for his 1937 mathematical paper titled "On Computable Numbers." The paper was an attempt to identify Gödel's undecidable questions; that is, questions that were beyond the reach of logical proof. In order to identify these undecidable questions, Turing proposed a universal Turing machine whose internal working could be altered so that it could perform different functions. In effect, Turing created the blueprint for the modern programmable computer.[2]

Turing joined Bletchley Park in September 1939 and focused his work on how to crack the Enigma cipher without using the message-key-repetition loophole. In effect, Turing was working to handle the scenario where the Germans would rectify this loophole in the Enigma cipher. Towards this aim, he studied the vast collection of captured messages and the corresponding decrypted plaintext that Bletchley had accumulated in the past. Turing noticed that most of the messages sent by the Germans conformed to a certain structure. Using this knowledge, along with other aspects of the message like the time, source and destination of the message, Turing realized he could sometimes get enough information about the message to predict parts of the message. Turing used these "cribs" (ciphertext-plaintext pairs) along with his background in mathematics and science to build his version of the bombes, which

[2] To be fair to history, the universal Turing machine was a reincarnation of Charles Babbage's Difference Engine No. 2.

could crack the new complicated version of the Enigma cipher. The effectiveness and success of Turing's bombes can be estimated by the fact that a bombe could discover a message key in less than an hour, and there were as many as 49 bombes employed in Bletchley Park by the end of 1942.

Enigma was by no means the only cipher of World War II. There were many other ciphers used by the Germans and the Allied forces, cryptographers and cryptanalysts. Breakthroughs were achieved on either side of the fence, but perhaps no story is as interesting (or probably as well documented) as the story of the Enigma.

Moving On

Now that we have a historical perspective of wireless technology and cryptography, we can look at the technical aspects of wireless security but before we begin, I would like the reader to be aware of what to expect from this book.

Wireless Security is a vast topic and any attempt to address all issues in a single book is a daunting task. It is almost impossible to explain each and every security algorithm that is used in wireless security in detail in a single volume. In writing this book, I have tried to strike a balance between architectural overviews and minute details. The aim of this book has been to answer questions like: How is wireless security different from security of wireline networks? How has wireless security evolved with changes in wireless networking? What is the architectural philosophy behind the design of wireless security protocols? What are the loopholes in these protocols and how can they be rectified in future designs? With this aim in mind, the rest of this book is organized as follows:

Chapter 1 looks at the basics of cryptography.

Chapter 2 delves into how cryptography is used to provide network security. We also look at some of the most prominent security protocols used in network security today. We do not distinguish between wireline and wireless networks at this stage of the book.

Chapter 3 explores the topic of security and the layered-network architecture. This is an often-neglected topic and one which, I think, is necessary to understand the architectural integration of security and networking.

Chapter 4 lays the foundation of wireless networks and discusses the design of cellular networks which were primarily designed for enabling voice communication.

Chapter 5 discusses wireless networks which are primarily designed for enabling data communication. These networks are a relatively new phenomenon and probably the most widely deployed technology in this field is IEEE's 802.11. We look at 802.11 networks and also explore mobile ad hoc networking which is an active area of research today in wireless networking.

Chapter 6 looks at the evolution of security in cellular networks. We start with security implementation in the first generation (AMPS) cellular networks and see how security implementations evolved and improved in second generation (GSM) and then third generation (UMTS) cellular networks.

Chapter 7 studies security in 802.11 networks. This topic has been a topic of hot debate in the recent past. We look at the security implementation of 802.11 in detail and study what all the debate was about. We then look at how the 802.11i specifications aim to fix the loopholes of 802.11 networks.

Finally in Chapter 8, we look at security in wireless ad hoc networks. Ad hoc networks (and specifically multihop ad hoc networks) are an active area of research today. Needless to say, then, that security in such networks is also being actively researched. This chapter looks at the underlying security concepts which seem promising for securing ad hoc networks.

Reading Conventions

Alice and Bob are the common archetypal characters used in cryptography/security. I have stuck to this convention where Alice and Bob want to communicate with each other securely over an unsecure network. Eve is an eavesdropper/attacker who wishes to eavesdrop on their communication or disrupt their communication or launch other sorts of attacks. These three characters would be sufficient for explanation of the text. Another convention to keep in mind is that the encryption of a message (M) using a key (K) is depicted either by EK(M) or K(M) whereas the hash of a message (M) using a key (K) is depicted by HK(M).

How to Read this Book?

Begin at the beginning and go on till you come to the end; then stop.

—Lewis Carroll, *Alice in Wonderland.*

I cannot possibly say it better than Lewis Carroll: I would like every reader to read the whole book, but since that is not always possible, I have tried to organize this book in as modular a fashion as possible to allow maximum flexibility; for example, readers interested only in cellular networks security may skip Chapters 5, 7 and 8. On the other hand, those with a background in cryptography may skip Chapters 1 and 2.

In the end, it is up to the reader. Anyway you read it, I hope this book helps you in understanding security in wireless networks.

Acknowledgments

I am thankful to my father for teaching me the values and morals of life. I am thankful to my mother who never had time to do anything else but be a mother to me and my sister. I am thankful to Sumedha for being a loving sister and for always being there. I am thankful to my wife for her love, support and constant encouragement without which this book would have never been written. I am thankful to her for putting up with my long hours on the computer and also for being the proof-reader, first-cut editor, typesetter and typist for this book.

I am also thankful to the following people:

David Lide – my numerous discussions with who have improved my understanding on various technical topics.

Chandra Didi, Jijaji and the Ahuja Family – for their encouragement and support.

Ashwin Mahajan – my discussions with who taught me lateral thinking.

Ankur Bhagat – from whom I learnt the meaning of perseverance.

and

Harry Helms, my editor – whose encouragement and guidance were invaluable in writing this book.

Finally, I would like to thank all the authors whose text has been listed in the References section. Many references were used for this book but some may have been mistakenly left out from the Reference section: to all these authors who recognize their words and ideas represented here but do not see their work in the reference section, my apologies.

Acronyms

Symbol

1G	First Generation
2G	Second Generation
3G	Third Generation

A

ACO	Authentication Ciphering Offset
AES	Advanced Encryption Standard
AH	Authentication Header
AK	Anonymity Key
AKC	Asymmetric Key Cryptography
AMF	Authentication Management Field
AMPS	Advanced Mobile Phone System
AODV	Ad Hoc On-demand Distance Vector
AP	Access Point
ARAN	Authenticated Routing for Ad Hoc Networks
ARP	Address Resolution Protocol
AuC	Authentication Center

B

BS	Base Station
BSA	Basic Service Area
BSC	Base Station Controller
BSIC	Base Transceiver Station Identity Code
BSS	Base Station System
BSS	Basic Service Set
BTS	Base Transceiver Station

C

CA	Certificate Authority
CBC	Cipher Block Chaining

CCMP	Counter Mode CBC-MAC Protocol
CCS	Common Channel Signaling
CD	Collision Detection
CDMA	Code Division Multiple Access
CEPT	Conference of European Postal and Telecommunication
CHAP	Challenge Handshape Authentication Protocol
CI	Cell Identifier
CO	Central Office
CRC	Cyclic Redundancy Check
CRL	Certification Revocation List
CSMA-CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear To Send

D

DAD	Duplicate Address Detection
DCC	Digital Color Code
DDoS	Distributed Denial of Service
DES	Digital Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DIFS	Distributed Inter-Frame Space
DoS	Denial of Service
DS	Distribution System
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
DSSS	Direct Sequence Spread Spectrum

E

EAP	Extensible Authentication Protocol
EAPoL	Extensible Authentication Protocol over Lan
ECB	Electronic Codebook
EIR	Equipment Identity Register
ELP	Equational Logic Programming
ESN	Electronic Serial Number
ESP	Encapsulating Security Payload
ESS	Extended Service Set

F

FHSS	Frequency Hopping Spread Spectrum
FMS	Fluhrer-Mantin-Shamir

G

GAP	Generic Access Profile
GFSK	Gaussian Frequency Shift Keying
GMSC	Gateway Mobile Switching Center
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GSM	Global Systems for Mobile Communications

H

HLR	Home Location Register
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol

I

IANA	Internet Assigned Numbers Authority
IBSS	Independent Basic Service Set
ICV	Integrity Check Value
IK	Initialization Key
IKE	Internet Key Exchange
IMEI	International Mobile Station Equipment Identity
IMSI	International Mobile Subscriber Identity
IPSec	Internet Protocol Security
IS41	Interim Standard 41
ISP	Internet Service Provider
IV	Initialization Vector

K

KAC	Key Administration Center
KDC	Key Distribution Center
KSI	Key Set Identifier

L

LAN	Local Area Network
LK	Link Key
LMSI	Local Mobile Subscriber Identity
LS	Land Station

M

MAC	Media Access Control
MAC	Message Authentication Code

MANET	Mobile Ad Hoc Network
MANET	Multihop Ad Hoc Network
MAP	Mobile Application Part
ME	Mobile Equipment
MIC	Message Integrity Check
MIN	Mobile Identification Number
MK	Master Key
MPDU	Media Access Control Protocol Data Unit
MS	Mobile Station
MSC	Mobile Switching Center
MSDU	Media Access Control Service Data Unit
MSISDN	Mobile Subscriber ISDN Number
MSRN	Mobile Station Roaming Number
MTSO	Mobile Telephone Switching Office

N

NAT	Network Address Translation
NAV	Network Allocation Vector
NIST	National Institute of Standards and Technology

O

OFB	Output Feedback
OFDM	Orthogonal-Frequency-Division-Multiplexing
OLSR	Optimized Link State Routing
OOB	Out-Of-Band
OSA	Open System Authentication
OSI	Open Systems Interconnection

P

PAN	Personal Area Network
PAP	Password Authentication Protocol
PDA	Personal Digital Assistant
PHY	Physical Layer
PK	Payload Key
PKC	Public Key Cryptography
PKEY	Pass-Key
PKI	Public Key Infrastructure
PLCP	Physical Layer Convergence Protocol
PMD	Physical Medium Dependent
PMK	Pair-wise Master Key

PN	Packet Number
PoP	Point of Presence
PPP	Point to Point Protocol
PRF	Pseudorandom Function
PSTN	Public Switched Telephone Network
PTK	Pair-wise Transient Key

Q

QoS	Quality of Service

R

RADIUS	Remote Access Dial In User Security
RF	Radio Frequency
RNC	Radio Network Controller
RSA	Rivest-Shamir-Adleman
RSN	Robust Security Network
RSS	Received Signal Strength
RTS	Request To Send

S

SA	Security Association
SAR	Security-aware Ad Hoc Routing
SAT	Supervisory Audio Tone
SCM	Station Class Mark
SEG	Security Gateway
SID	System Identifier
SIFS	Short Inter-Frame Space
SIG	Special Interest Group
SIM	Subscriber Identity Module
SKA	Shared Key Authentication
SKC	Symmetric Key Cryptography
SP	Signaling Points
SPI	Security Parameter Index
SQN	Sequence Number
SS7	Signaling System #7
SSID	Service Set Identifier
SSL	Secure Sockets Layer
SSP	Service Switching Point
STA	Station
STP	Signaling Transfer Point

T

TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TGT	Ticket Granting Ticket
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
TMSI	Temporary Mobile Subscriber Identity
TSC	Tkip Sequence Counter
TSN	Transitional Security Network
TWN	Traditional Wireless Network

U

UEA	User Encryption Algorithm
UIA	User Integrity Algorithm
UMTS	Universal Mobile Telecommunications System
URL	Universal Resource Locator

V

VLR	Visitor Location Register
VPN	Virtual Private Network

W

WAE	Wireless Application Environment
WAN	Wide Area Network
WAP	Wireless Application Protocol
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WPA	Wi-Fi Protected Access
WTLS	Wireless Transport Layer Security

Z

ZRP	Zone Routing Protocol

About the Author

Praphul Chandra is an Electrical Engineer by training. He completed his undergraduate studies from Institute of Technology, BHU in India and his graduate studies from Columbia University, New York. He works for Texas Instruments and lives with his wife in Maryland. This is his first book. He is currently working on his second book which would be about Wi-Fi Telephony. He maintains his personal website at www.thecofi.net.

CHAPTER 1

Security and Cryptography

1.1 What is Security?

"When I use a word," Humpty Dumpty said, in rather a scornful tone, "it means just what I choose it to mean—neither more nor less."

—Through the Looking Glass.

The Webster dictionary defines security as *the condition or quality of being free from apprehension, anxiety, or care*. A secure communication network, then, can be defined as a network whose users do not feel any apprehension or anxiety while using the network.

Note how the meaning of a secure network depends on how it is used. As an example, consider the Internet. As long as the Internet was the domain of engineers and scientists, users did not care about security. If somebody was geeky enough to connect to the Internet, they had the right to use it anyway they wanted. With the commercialization of the Internet came a big boom in the number of users. Along with this boom, however, came security concerns. These security concerns grew manifold with the coming of age of e-commerce. Today, users use their credit card over the Internet as much as they use it over the telephone. What are the security expectations of such users?

Let's consider this with a concrete example. Suppose Alice (A) wants to carry out a monetary transaction with her Bank (B). First and foremost, A expects that the information she intends to send to (or receive from) B should not be accessible to anybody

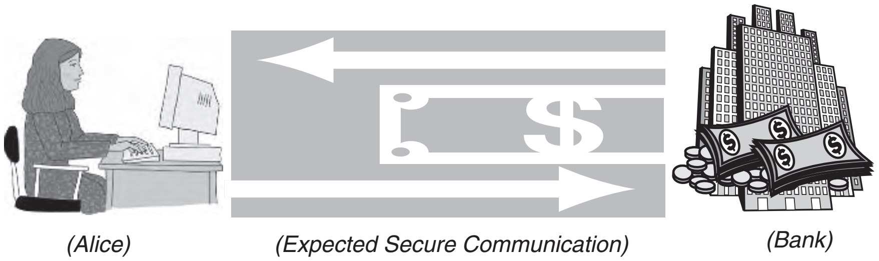


Figure 1.1: The Need for Security

else. This is the expectancy of confidentiality. Next, she expects that the information she sends to (or receives from) B should not be altered or corrupted by anybody. This is the expectancy of integrity. Note that confidentiality and integrity are mutually exclusive principles in that the existence of one does not imply the existence (or the absence) of the other.[1]

Since B too is a user of the underlying network, we should consider also, the security expectations of B. First, B expects that A is who she says she is. After all, you would not expect your bank to let someone else operate or access your account. This is the expectation of authentication. B also expects to have proof of the transaction that A carries out so that A cannot deny having withdrawn money from her account in the future. This is the expectation of nonrepudiation.

Both A and B also expect that they would be able to carry out transactions without losing connection or communication with each other. At its face value, this appears to be a networking problem rather than a security problem. However, Denial of Service (DoS) attacks in the recent past have exploited weaknesses in network security protocols to completely bring down servers and networks, thus bringing this problem into the security domain.

To summarize, a secure communication network provides the following facilities to its users:

Confidentiality: The non-occurrence of the unauthorized disclosure of information. No one except the sender and the receiver should have access to the information being exchanged.

Integrity: The non-occurrence of the unauthorized manipulation of information. No one except the sender and the receiver should be able to modify the information being exchanged.

Authentication: The receiver's ability to ascertain the origin of a message. An intruder should not be able to masquerade as someone else.

Nonrepudiation: The receiver's ability to prove that the sender did in fact send a given message. The sender should not be able to falsely deny later that he sent a message.

Service Reliability: The ability to protect the communication session against denial of service attacks.

[1] If the message is encrypted, an intruder may have access to the data being sent but does not have access to the "information" being sent. In this case, an intruder may still be able to modify the data, leading to a modification in the information.

It is extremely important to realize that these are distinct independent requirements and the presence or absence of any one of them does not in any way guarantee the presence or the absence of the other(s).

1.2 Cryptography

Now that we understand the requirements of a secure communication network, the next question is how to satisfy these requirements. Here is where cryptography comes in. Cryptography is the art and science of keeping messages secure[1]. Let us look at how cryptography makes messages "secure" in the context of our requirements of secure communication networks.

1.2.1 Confidentiality

Users of a network use human languages (like English, Hindi, French, and so forth) to communicate over a network. Let us suppose that A uses English for this transaction. The messages that are exchanged between A and B are therefore in plain English or *plaintext*. Consider now that A is sitting at her home in our example and that the media (twisted-pair telephone wires, coax cable, wireless, and so forth) which carry her messages go beyond her home boundaries. This means that anybody may access the media without A's knowledge.

Now, let us introduce Eve (E), the eavesdropper (or a private investigator, if you want to keep things exciting) who wants to keep track of A's financial transactions. All E needs to do to access the messages is to access the media. Since the messages exchanged between A and B are in plaintext, E will have no problem accessing the information she needs if she can access the media.

Let us suppose that A wants her messages to be confidential. To do so, she needs to send out the messages in a format that E will not understand. Put another way, A wants to send messages to B in secret or ciphertext. The process of converting plaintext messages into ciphertext is called encryption. Encryption can therefore be defined as the process of disguising a message so as to hide its substance or the information contained in the message. Conversely, decryption is the process of obtaining the plaintext from the ciphertext and can therefore be defined as the process of obtaining the information contained in an encrypted message. Mathematically, if M represents the plaintext message and C represents the ciphertext message, then we can say:

Encryption :: $E(M) = C$
Decryption :: $D(C) = M$.

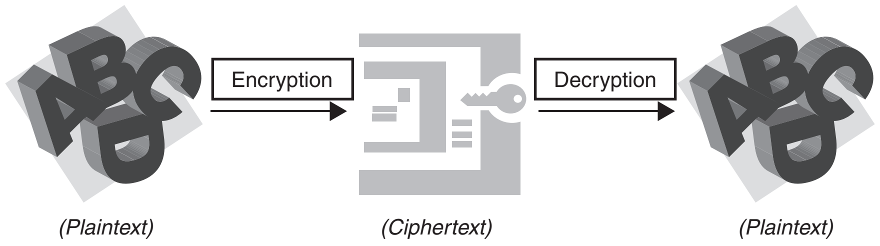


Figure 1.2: Cryptography in Action

The assumption we have made in this example is that E is incapable of decrypting the ciphertext to obtain the information. To be more precise, the assumption is that the messages sent by A to B would be secure if B and only B knows how to decrypt messages that A is encrypting. Looked at another way, this means that A and B share a secret—that is, how to encrypt and decrypt messages. So we can say that confidentiality here is achieved by sharing of a secret. This secret is the method (algorithm/cipher) A uses to encrypt and B uses to decrypt messages.

There are practical limitations for using this methodology. First, this scheme of things require that A and B have a predetermined algorithm to use. This assumes the existence of a secure channel between A and B (a face-to-face meeting, a secure telephone conversation, and so forth). This is not always possible. Second, B must have a separate algorithm for each of its customers, otherwise one customer may be able to eavesdrop on the transactions of other customers. This requires B to be able to use a huge number of algorithms. Not only does this increase the system complexity for B, it also places an additional burden on the system designers to find so many cryptographic algorithms.

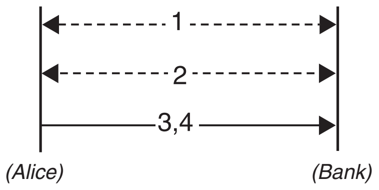
Note further that in this scheme of things, the confidentiality of the system is enforced by sharing the cipher (encryption/decryption algorithm) and keeping it a secret between the parties involved. Mathematically, such a security system is represented by D(E(M)) = M. However, such a system has severe cryptographic limitations. Suppose that there is a group of people who want to communicate with each other over a medium. Now, while communicating, two people want to ensure the confidentiality of the messages between them—that is, they do not want others in the group to be able to access the messages they are exchanging. Since the cipher is known to all group members, two-party confidentiality cannot be achieved by this security system. This is where keys come in. A key is a shared secret between communicating parties which can be used for securing communication between them.

When a cipher uses a key to encrypt and decrypt messages, the secret to be shared moves from the algorithm to a number (key). We can now achieve secure communication between any number of nodes in an unsecure environment by ensuring that the participating nodes share a secret key. In other words, the security in these systems is based on the key and not on the details of the algorithm. This is also much easier to implement, since keeping a number secret is much more easier than keeping an algorithm secret. A key is one of a large number of values. The set of values from which the key can be selected is known as the keyspace. The larger the keyspace, the more secure is the key since the key would be harder to guess (break).

1.2.1.1 Symmetric Key Cryptography

Keys can be used symmetrically or asymmetrically. Symmetric Key Cryptography (SKC) refers to the process wherein the sender and the receiver use the same shared key (and the same cipher) to encrypt and decrypt messages. Such a system is represented mathematically as $D_K(E_K(M)) = M$.

In our example, suppose A and B decide to use SKC to maintain the confidentiality of their transaction. Assuming the real life scenario, wherein A and B have no preestablished standards or knowledge about each other, establishing secure communication between them requires the following steps:



1. A and B agree on a cryptosystem (cipher to be used).
2. A and B agree on the key to be used.
3. A encrypts messages using the negotiated key and cipher and sends them to B.
4. B decrypts the ciphertext using the negotiated key and cipher.

Figure 1.3: Symmetric Key Cryptography

The Achilles heel of symmetric key cryptography is that step 2 has to be carried out before the underlying channel has been made secure.

Step 2 being carried out unsecurely means that if Eve gets access to the channel during this step, she can know the key that A and B decide to use. This compromises the security of the system. In synopsis therefore, symmetric key cryptography has the following problems:

- Keys need to be distributed in secret, since the security of the system lies in the secrecy of the keys. Since there is no inherent support for key distribution, keys are usually distributed by some other means (like couriers, for example). This means that the system is as secure as the courier. Furthermore, key distribution can be a daunting task for large networks like the Internet.
- Once the key is stolen, the whole security system breaks down—there is no graceful death.
- Assuming a separate key is used for each pair of nodes, the total number of keys required for an n-node network is n(n – 1)/2. This quantity grows dramatically with increasing n.

1.2.1.2 Asymmetric Key Cryptography

Asymmetric Key Cryptography (AKC) exploits the mathematics of trapdoor one-way functions. A one-way function is a function *f* such that it is easy to compute *f*(*x*) for any given *x* but it is very hard to compute *x* for a given *f*(*x*). "Very hard" usually means that it takes a very long time with current computer power standards to calculate *x* from *f*(*x*). An example of a one-way function is the square of a number. It is easy to compute the square of a number but it is much harder to compute the root of a number. However, calculating the roots is not hard enough to be used in cryptography.

A trap-door one-way function is a one-way function with the peculiarity that it becomes easy to calculate *x* from *f*(*x*) if and only if we know a secret key. In other words, *x* to *f*(*x*) is easy and *f*(*x*) to *x* is hard except if we know the secret *y*; in which case, given *y* and *f*(*x*), it becomes easy to get *x*. An example of trap-door one way function is the product of two large prime numbers. Such a product is easy to compute but given the product it is very hard to compute its factors and obtain back the numbers. In fact the larger the prime numbers the more hard it becomes to factor the product. However, the trapdoor is that if we know one of the prime numbers then the problem becomes the trivial problem of division to obtain the other number. So, one of the primes serves as the key here.

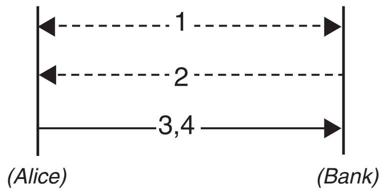
Asymmetric key cryptography refers to the process wherein the sender and the receiver use different keys (but the same cipher/algorithm) to encrypt and decrypt messages. Such a system is represented mathematically as $E_{K1}(M) = C$ and $D_{K2}(C) = M$ or

$$x \rightarrow f(x) :: \text{Easy}$$
$$f(x) \rightarrow x :: \text{Very Difficult}$$
$$f(x) + Y \rightarrow x :: \text{Easy}$$

Figure 1.4: Trap Door One-Way Functions

$D_{K2}(E_{K1}(M)) = M$. The decryption key is different from the encryption key and should not be derivable from the encryption key. Asymmetric key cryptography is also called Public Key Cryptography (PKC) since either the encryption key or the decryption key may be made public depending on whether we need confidentiality or nonrepudiation.

To achieve confidentiality using PKC in our example of Alice and the Bank, we use the mechanism described in Figure 1.5.



1. A and B agree on a cryptosystem (cipher to be used).
2. B sends its public key to A.
3. A encrypts messages using the negotiated cipher and B's public key received in step 2.
4. B decrypts the ciphertext using its private key and the negotiated cipher.

Figure 1.5: Public Key Cryptography

The strength of PKC lies in the fact that the steps carried out in the unsecure channel (steps 1 and 2) do not compromise the security of the system. This is so because it is the encryption key which is sent over the unsecure medium and made public. This means that whoever gets access to the public key (say Eve) can encrypt messages and send them to B but Eve cannot decrypt messages meant for B since the private key is still a secret. Therefore when PKC is used to achieve confidentiality, the encryption key is the public key and the decryption key is the private key.[2]

The secrecy of the system lies in the decryption key, which is never transmitted over the medium. The encryption key is made public by the receiver so that anyone can send messages to them securely. Since only the receiver has the decryption key, only he can decrypt the message.

To summarize, B maintains a pair of keys (K1,K2) per user. When a user (Alice) wants to make a transaction, B sends her the encryption key K1 over the unsecure medium. Alice encrypts all her messages using this key ($E_{K1}(M)$) and sends the resulting ciphertext over the media. Now, even though E has access to the ciphertext, C and

[2] On the other hand, the decryption key is made public by the sender to implement digital signatures. In digital signatures, since only sender can encrypt the message, anyone who receives the message (and decrypts it using the public key) can be ensured that the message is authentic.

the encryption key, K1, she cannot access the information contained in the messages since $D_{K1}(E_{K1}(M))$ does not yield M.

Mathematically, access to $f(x)$ (the ciphertext) does not help rogue nodes since it is "very hard" to obtain x from $f(x)$. This happens because f is a trap-door one-way function. The trap-door in this case is the private key, K2 which never leaves B. Since the private key, K2 (the trap-door) is known only to the destination of the message (B), only B can decrypt the message.

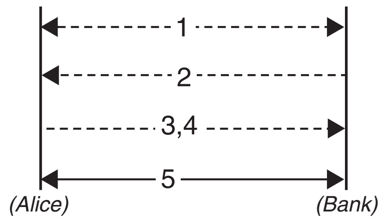
Ideally, we would always like to use PKC. However, everything comes at a cost. PKC algorithms are slow. In fact, they are almost 1000 times slower than SKC algorithms. Though this may not seem a major hindrance given the increasing processing power and speeds of today's computers, we should remember that as the speed of PKC grows, so would the speed of SKC, keeping the latter an attractive option for real time applications like voice. Also, with time, the amount of data being transmitted per application will grow with the growth in multimedia networking.

Note that the above solution achieves confidentiality in one direction. In other words, E would be unable to read messages that A sends to B but E can read messages that B sends to A. To achieve confidentiality in both directions, A would need to maintain a pair of keys, send the encryption key to B and then decrypt messages it receives from B using the corresponding pair-key.

It is important to point out here that cryptography is not a perfect solution (not today, at least). Even PKC is susceptible to chosen plaintext attacks. These attacks exploit the fact that the encryption key is made public in PKC. Here is how a rogue person may break into a system using PKC. First, the rogue person learns the public key of a node and then encrypts all possible (or all probable) plaintext messages that it expects the node to receive. Note that this can be done off-line. Once finished, the rogue node has a one-to-one mapping of the ciphertext messages and the plaintext messages. It can then intercept ciphertext messages in real time, hash them into the mapping and obtain the corresponding plaintext. Such an attack is highly effective if the set of plaintext messages that the node receives are limited. This happens, for example, in ATM machines communicating with the central database.

1.2.1.3 Hybrid Cryptography

Hybrid systems exploit the speed of SKC and use PKC to solve the key distribution problem of SKC. Refer to Figure 1.6 to see how A and B communicate securely in a hybrid cryptosystem.



1. A and B agree on a cryptosystem (cipher to be used).
2. B sends its public key to A. Let's call this key K1.
3. A generates a random session key, K. It encrypts the session key K with B's public key and sends it to B, i.e., A transmits $E_{k1}(K)$.
4. B decrypts the received message using its private key to obtain the session key, K. At this point both A and B (and only A and B) know the session key to be used for this transaction.
5. Both A and B use the session key for this session to communicate securely.

Figure 1.6: Hybrid Cryptography

As seen above, hybrid cryptography uses PKC for key distribution but SKC for message encryption. Since key distribution is done once per session and this removes the burden of key distribution, the trade-off is worth it. SKC for message encryption fulfils the speed requirements of real time applications. Note also that this method uses a distinct key for every session and since this session key is generated at runtime, it drastically reduces the probability of compromising the key's security. Another advantage of this scheme is that it automatically achieves confidentiality in both directions.

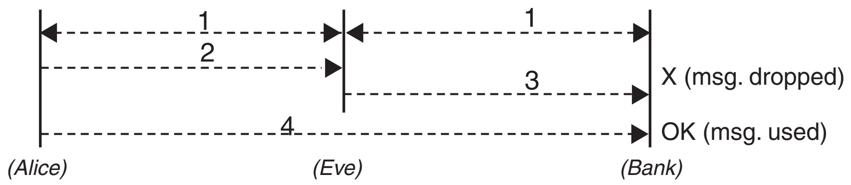
1.2.2 Integrity

Suppose that A is sending encrypted messages to B. When Eve gets access to the transmission medium between A and B, she starts accessing the messages that A is sending to B over the medium. Although Eve (E) is not able to make any sense of the packets that she is intercepting (since the messages are encrypted), E randomly starts modifying these messages on the fly. When B receives the messages, he has no way of knowing that the message has been tampered with, so he continues to process these messages. On decrypting the ciphertext, B may either get a message different from what A had sent, or some garbled message. Either is undesirable since it violates the integrity of the messages.

So, how does cryptography satisfy the integrity requirement of a communication network? For the receiver to ensure that the message sent by the source has not been modified or tampered with, cryptography exploits one-way hash functions. These are one way functions with the additional property that they take an input of variable length but produce an output of fixed (and much shorter) length. For cryptography, there are two desirable features in one-way hash functions. First, a small change in the message should produce a large change in the hash of the message. This helps because a rogue person trying to tamper with the message would likely change only a small piece of information in the message. Second, the hash function should be collision-free. This means that the probability of two different messages producing the same hash should be very low. This property protects against the attack wherein the rogue node tries to completely change the message while trying to maintain the same hash.

Figure 1.7: Message Authentication Code

In cryptography, the sender of the message calculates the hash of the message. This hash is known by various names like Message Authentication Code (MAC), message digest, fingerprint, message integrity check, and so on. Now, if Eve modifies the message (and/or the MAC), B will know that the message has been tampered with. Refer to Figure 1.8 to see how this works:



1. A and B negotiate the one-way-hash function they will use. E eavesdrops on this information.
2. Before sending a message, A (or B) computes the hash of the message and attaches it to the message.
3. Eve modifies this message and sends it to B. When B calculates the MAC of the message and compares it to the attached MAC, the two do not match and B, therefore, knows that the message was tampered with. B, therefore, drops this message.
4. If the message is un-tampered, B accepts the message since the MAC he calculates and the MAC attached to the message are identical.

Figure 1.8: Using MAC for Integrity

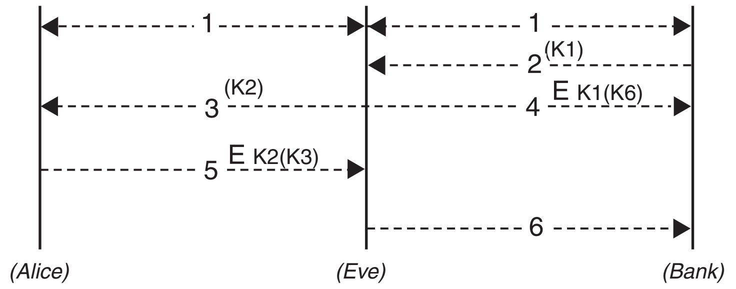
The strength of the scheme lies in Eve's inability to modify the message without modifying the corresponding MAC. This happens because a small change in the message produces a large change in the hash of the message and because the probability of the modified message having the same MAC as the old message is very low. Consider also the fact that Eve might try to create totally new messages on her own, attach her own MAC and try to pass them off as Alice's messages, but this is an authentication problem and is explained in the next section.

It is possible to ensure that only the intended receiver of the message is able to calculate the MAC of the message. This is done by calculating the hash on the message content and a secret key shared between the sender and the receiver (obtained, for example, by PKC for SKC). In other words, the one-way hash function operates on a combination of the message content and the key to produce a hash which is known as the Message Authentication Code (MAC). Since an eavesdropper does not have the key, they cannot calculate the MAC of the message.

1.2.3 Authentication

Suppose that A and B are trying to communicate securely using hybrid cryptography for confidentiality and MACs for message integrity. Now, suppose Eve plans to rob bank customers by masquerading as them. Obviously, to protect against impostors like Eve, the bank needs a security feature. Authentication refers to the ability of a receiver to ascertain the origin of a message. This ensures that an impostor like Eve is not able to masquerade as someone else. Figure 1.9 shows why a simple "username-password" scheme is not sufficient to address this issue.

In cryptography, authentication is achieved by using digital signatures or challenge-response schemes. The former involves the sender digitally signing the message. If it



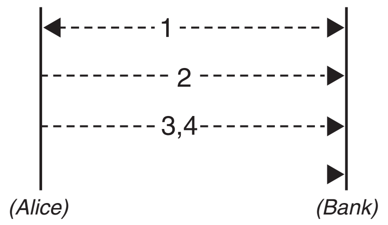
Hint: Odd numbered keys are real. Even numbered keys are used by E for fraud.

1. When A and B agree on a cryptosystem (algorithm to be used), E notes this down.
2. B sends its public key (K1) to A. E notes this down but does not send this to A.
3. E generates an encryption-decryption key pair (K2,K4) and sends the encryption key, K2 to A.
4. E generates a random key (K6) on her own, encrypts it using K1 and sends it to B. Note that B has no way of finding out that the message came from E and not A; so B proceeds to the next logical step of prompting for username and password before allowing access into the account.
5. Meanwhile, since A too has no way to find out that message it received came from E and not B, A proceeds to generate a random session key, K3. It encrypts K3 with K2 and sends it to B. However, E intercepts this, decrypts it using K4. At this time E has successfully tricked A into believing that A has a trusted session with B. So, B prompts A to send her username and password using the session key K3.
6. At this point in time, E knows the username and password of A. She sends this to B.

So, finally E has tricked A into believing that she is B and has tricked B into believing that she is A. She is practica
the master of the universe here.

Figure 1.9: Man-in-the-Middle Attack

is ensured that digital signatures cannot be copied or faked, the receiver can check for digital signatures on the messages to authenticate the message. Digital signatures use PKC in reverse. Refer to Figure 1.10 to see how digital signatures work.

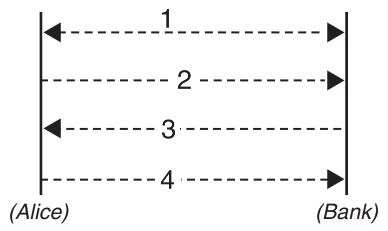


1. A and B agree on a cryptosystem (cipher to be used).
2. A sends its public key to B. Let's call this key K1.
3. A encrypts the message it sends to B with its own private key.
4. B decrypts the received message using K1 (A's public key). If B is able to successfully decrypt the message, B can be sure that it is talking to A since only A has its own private key.

Figure 1.10: Digital Signatures

Practical implementations of authentication do not involve encrypting the whole message with the private key, since this may take a long time. Instead, they encrypt the hash of the function. Since the hash of the message is much smaller than the message itself, this saves a lot of time. The receiver can then decrypt the hash of the message with the sender's public key, compute the hash of the message and compare the two to authenticate the message.

In the challenge-response scheme, B sends a random number to A. This is the challenge. A 'signs' this random number with her private key and sends this signed response back to the initiator. This is the response. B verifies this signed response by



1. A and B agree on a cryptosystem (cipher to be used).
2. A sends its public key to B. Let's call this key K1.
3. B sends a random number to A.
4. A encrypts this random number with its own private key and sends this to B. B decrypts the received message using K1 (A's public key). If B finds that it is successfully able to decrypt and obtain the random number it sent, B is ensured that it is talking to A.

Figure 1.11: Challenge-Response

decrypting the response. If this is verified, B authenticates A. A may also authenticate B by sending a challenge.

Challenge-response systems and digital signatures are variations of the same technique. Whereas the former requires a single separate message to ensure authentication, the latter does it on a per-message basis. Even though the latter appears to be more secure, the former one is well suited for connection-oriented networks where the authentication can be done once at connection setup time, thus saving the per-message over-head.

1.2.4 Replay Protection and Nonrepudiation

In the real world, when A gives a cheque to B, B needs to deposit the cheque in the bank to cash it. Since the bank takes possession of the cheque after cashing it, this ensures that B does not cash the cheque more than once. Now, consider a digital cheque. Suppose A sent B a digitally signed digital cheque. How do we prevent B from cashing the cheque more than once? Since there is no physical entity, the bank cannot take possession of something.

In the digital world, protection against reuse of an entity meant for one-time use is referred to as replay protection. In other words, we want to ensure that B can only use the cheque once. This is achieved by having A include a timestamp in the message, hash the message and encrypt it with its private key. Basically, the authentication mechanism spreads its scope to include a timestamp in the digital signature. This way two cheques sent to B will have two different digital signatures, even if everything else is exactly the same.

Timestamps also help solve the problem of nonrepudiation. Suppose A sends a cheque to B and later claims that she did not send it. The proof that B has is that the message he received was signed using A's private key and therefore A sent it. However, suppose A repudiates, claiming that she has lost her private key. Timestamps limit the problem to some extent, but nonrepudiation is one of the toughest security parameters and usually requires the involvement of a trusted third party.

1.3 Cryptanalysis

The antithesis of cryptography is cryptanalysis. Whereas the former aims to hide the information content of a message from unauthorized entities, the latter aims to recover the information content of a message. Put another way, cryptanalysts aim to recover the plaintext from the ciphertext without having access to the shared secret keys. An attempted cryptanalysis is called an attack.

The simplest form of cryptanalysis is a brute force attack where given the ciphertext, the cryptanalyst tries every possible key one by one to decrypt it until they obtain a meaningful plaintext. This type of attack minimizes the data input requirements (all it needs is the ciphertext) and maximizes the system resource requirements (time and computing power).

As we move along this book, we will see various techniques of cryptanalysis. Network security is, in fact, an ongoing struggle between the cryptographers trying to provide security and cryptanalysts trying to break it.

1.4 Block Ciphers

Cryptographic algorithms are broadly divided into two categories depending on the type of data they expect as input. Block ciphers take blocks (usually 64 bits) of data as an input, whereas stream ciphers operate on a bit (or a byte) at time. Given a key, a block cipher always encrypts a given block of plaintext to the same ciphertext. On the other hand, given a key, a stream cipher will encrypt a given bit (or byte) of plaintext to a different ciphertext.

The building blocks of block ciphers are permutation and substitution. Let's say we are dealing with k-bit blocks; a substitution specifies, for each of the 2^k possible values of the input block, the k-bit output. So, basically we have a substitution table. Since it is impractical to manage such tables for 64-bit blocks, substitution is done for 8-bit blocks which are obtained by splitting up the input block. To specify a completely randomly chosen substitution for k-bit blocks would take about $k.2^k$ bits. A permutation specifies, for each of the k input bits, the output position to which it goes. To specify a completely randomly chosen permutation of the k-bits would take about k. log k.

The aim of permutation and substitution is to maximize diffusion and confusion. Diffusion is the property of ciphers which measures how many bits change in the ciphertext when a single bit is changed in the plaintext. Block ciphers usually have a large degree of diffusion, that is, changing 1 bit in a block of plaintext changes almost half the bits in the ciphertext block. Stream ciphers, on the other hand, have null diffusion since by definition they operate on 1 bit at a time. Confusion is the property of ciphers which hides the correlation between the plaintext and the ciphertext.

One of the most common ways to build a secret-key block cipher is to break the input into manageable sized chunks (say 8 bits), do a substitution on each small chunk, take the outputs of all substitutions and run them through a permuter that is as big as the input, which shuffles the bits around. Then the process (a round) is repeated so that each bit winds up as an input to each of the substitutions. The multiple rounds ensure that the cipher has enough diffusion. Because of these multiple rounds, such block ciphers are also known as iterated block ciphers.

Feistel networks are iterated block ciphers where the encryption algorithm is of the form: $L_i = R_{i-1}$ and $R_i = L_{i-1}$ XOR $f(R_{i-1}, K_i)$. The function *f*() can be as complicated as we like (it may even be a hash function) but any Feistel network is guaranteed to be reversible. Digital Encryption Standard (DES) is a special case of Feistel networks which uses a proprietary "*f*()" (also known as mangler function).

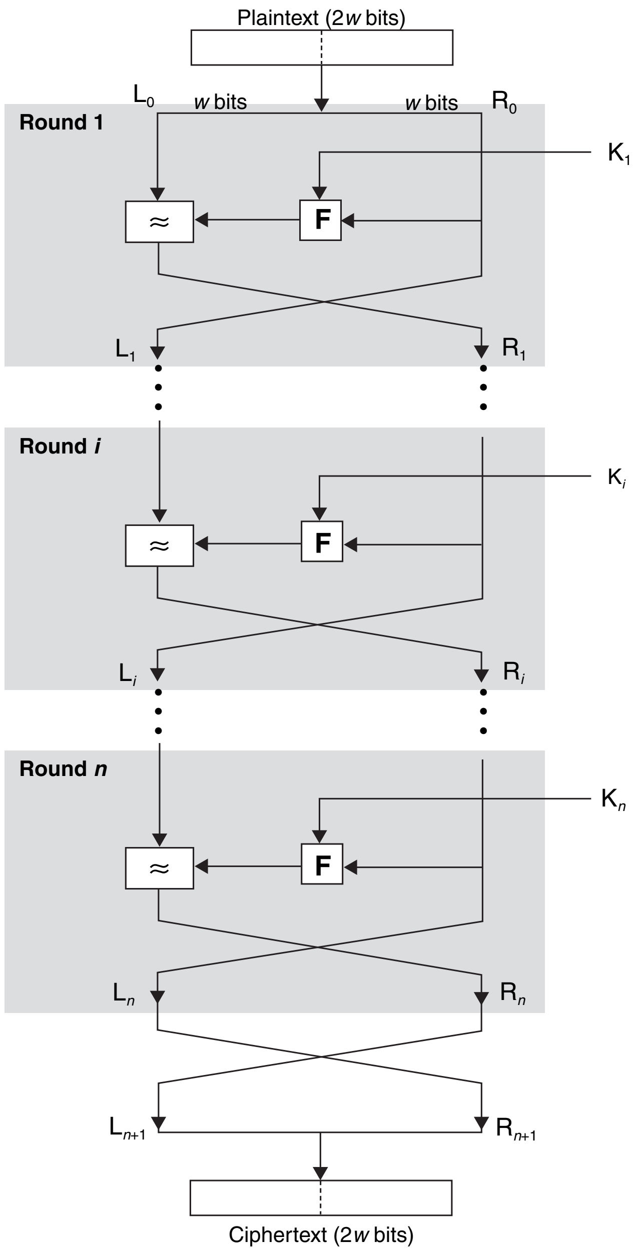


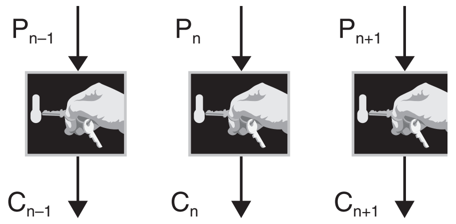
Figure 1.11a: A Feistel Network

1.4.1 Using the Ciphers: The Modes

Think of a cryptographic algorithm as a black box which takes some input and produces the encrypted message as the output. The input supplied to the algorithm consists of the plaintext message, the key(s) and some sort of feedback. The plaintext message is produced by the user. The keys will be dealt with in the next section.

This section will deal with the feedback issue and the "mode": a cryptographic mode refers to how the algorithm is used. The strength of the algorithm itself depends on the mode in which it is used. Different modes of an algorithm are suitable for various situations and needs, but the mode should not compromise the security of the underlying algorithm.

ECB: Electronic Codebook Mode (Block Ciphers):



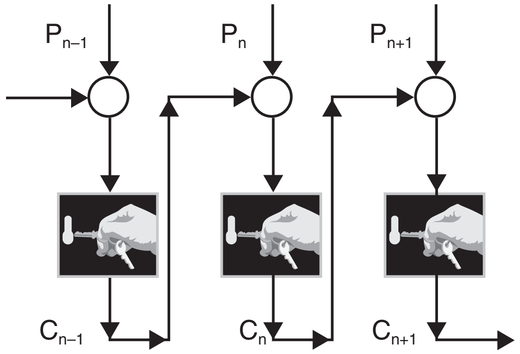
$C_i = E_k(P_i)$

Figure 1.12: ECB Mode

This is the simplest and the most obvious mode for block ciphers. Data is divided into 64-bit blocks and is fed to the cipher which produces the corresponding ciphertext. Each block is encrypted separately and therefore the operation can be carried out in parallel.

The advantage of this scheme is that it is simple, can be parallelized and there is no error propagation (a ciphertext block corrupted during transmission does not affect any other blocks). The biggest loophole in the scheme is that plaintext blocks are mapped one-to-one to ciphertext blocks. This means that repeats in plaintext blocks cause repeats in the ciphertext blocks. This property makes this mode vulnerable to splicing, re-ordering and replay attacks. In our example, Eve may create a <plaintext, ciphertext> directory and then reshuffle the cipher blocks in the message (or replay these blocks at a later time) to modify the message as she wants.

CBC: Cipher Block Chaining Mode (Block Ciphers):



$C_i = E_k(P_i \ XOR \ C_{i-1})$

Figure 1.13: CBC Mode

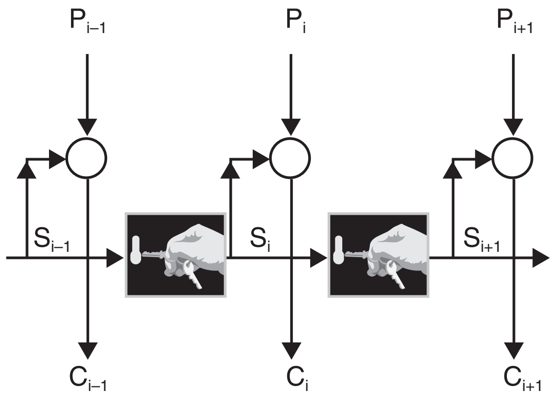
The basic problem with ECB is that a given plaintext always maps to the same ciphertext. We could solve this problem by XORing the plaintext with a random number (as large as the plaintext block) before encrypting it. If the random number changes for each block, the same plaintext will map to a different ciphertext each time it is encrypted. However, this scheme would require that Alice send the random number along with the ciphertext to Bob. This means that we double the traffic load on the network—not a very good idea.

CBC extends on the "random" number idea by using the ciphertext from the previous block as a "random number." Since Alice is transmitting the ciphertext from the previous block anyway, Bob already knows the "random number" being used and so there is no extra traffic to be transmitted. The chaining in CBC adds a feedback mechanism to the block cipher. The results of the encryption of the previous blocks are fed into the encryption of the current block. In other words, each block is used to modify the encryption of the next block and therefore each cipher block depends not only on the plaintext that generated it but on all previous plaintext blocks. This property makes it difficult for an attacker to decrypt selected ciphertext blocks. The attacker must have access to the complete session in order to decrypt any one cipher block.

Note that chaining forces identical plaintext to map to different ciphertext if and only if some previous plaintext in the session is different. This means that identical sessions still map to the same ciphertext session and two messages that begin the same (most packets have similar headers) will encrypt in the same way up to the first difference. To prevent this, the first block of data is fed a random Initialization Vector (IV). This IV may not be secret and may be sent to the receiver unsecurely. This does not compromise the security of the system.

The advantage of CBC is that it is much more secure than ECB and is self-recovering from errors since an error in C_i effects only P_i and P_{i+1}. The loophole in CBC is that flipping a bit in C_i flips the corresponding bit in $P_{i+1.}$ In our example, if Eve knows the message format that is used for the communication between Alice and Bob, she may flip a certain bit in a certain ciphertext block to change the message as she wishes. Note that even though flipping a bit in C_i will flip the corresponding bit in P_{i+1}, it will also garble P_i. However, this may not be a good enough indication for Bob to determine that the message has been tampered with. This shows why it is important to ensure the integrity of the message.

OFB: Output Feedback Mode (Stream Cipher):



$C_i = P_i \text{ XOR } S_i S_i = E_k(S_{i-1})$
$P_i = C_i \text{ XOR } S_i S_i = E_k(S_{i-1})$

Figure 1.14: OFB Mode

OFB is a method of using a block cipher as a stream cipher. We start with the block-size Initialization Vector (IV) and encrypt it using the key to obtain b_0, such as $S_0 = E_k(IV)$. Next we encrypt S_0 to obtain S_1 ($S_1 = E_k(S_0)$) and so on. Note that all this can be done off-line both on the sender and the receiver side since it does not require the presence of the plaintext or the ciphertext.

When the plaintext message arrives at the sender side, each block can simply be XORed with the appropriate S_i to obtain the ciphertext. On the receiver side, the ciphertext is again XORed with the appropriate S_i to obtain the plaintext back. Confidentiality, as always, is obtained by keeping the keys (used to generate the pseudorandom string) secret. Note that the feedback mechanism is independent of both the plaintext and the ciphertext and is used in the pseudorandom string generation.

The advantages of OFB are that it is fast (key-stream can be precomputed), does not require any padding and that errors in ciphertext cause limited errors in plaintext. The disadvantages are that it is not self-synchronizing and is susceptible to known plain-text attacks.

1.5 Stream Ciphers

"Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin."

—John von Neumann

Stream ciphers operate on plaintext 1 bit at a time and consist of two components: a key-stream generator and a mixing function. The mixing function is typically an XOR while the key-stream generator varies from one cipher to the other. Even though block ciphers are usually more secure than stream ciphers, the incentive for using stream ciphers lies in that they are faster and easier to analyze. The former property makes them more suitable for real time applications like voice communications and the latter property makes them more likeable by cryptographers who like to know how the system is secure rather than relying on black-box mangler functions and/or substitution boxes of block ciphers.

An optimal key-stream generator has a period of 2^N for an N-bit key. The period of a key-stream generator refers to the time that it takes for regenerating the same key-stream.

The security of the system lies entirely on the insides of the key-stream generator; for example, if the key-stream generator generates a stream of zeroes, the ciphertext is the plaintext. On the other hand, if the key-stream generator generates an endless stream of random bits, we have perfect security. For stream ciphers to work correctly, both the sender and the receiver must generate the exact same random number stream so that the sender can XOR it with the plaintext to obtain the ciphertext and the receiver can XOR it with the ciphertext to obtain the plaintext. However, there is an inherent contradiction in these two goals of generating random numbers perfectly and deterministically.

So, the key-stream generator at either end generates pseudorandom numbers. These pseudorandom numbers are used to encrypt data at the sender. The magic is that the key-stream generator at the receiver also generates the same pseudorandom number string flawlessly. Security is provided by making the output of the key-stream generator a function of a key provided to it as an input. The key should first be negotiated securely between the sender and the receiver (by using PKC for example). Then, the key-stream generator at the sender and the receiver use the negotiated key to produce identical output.

Pseudorandom numbers are numbers that "appear" to be random, but have a deterministic pattern since they are, after all, produced by some finite state machine. The more obscure this pattern (that is, the longer the repetition period), the better is the security provided. How "long" a period is long enough depends upon the application. However, it is imperative that the key-stream generator must have a period much larger than the number of bits the generator will output between key-exchanges. For a T-1 link which will encrypt 2^{37} bits per day, the period must be orders of magnitude larger than this even if the key is exchanged daily.

Stream ciphers can be divided into two categories (or modes). If the state of the key-stream generator depends on just the key, the cipher is called a synchronous stream cipher. If, on the other hand, the state of the key-stream generator depends on some previously produced ciphertext along with the key, the cipher is called a self-synchronizing cipher.

Synchronous Stream Ciphers:

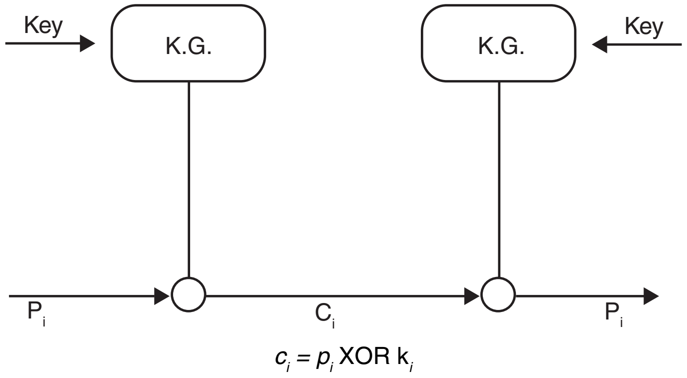


Figure 1.15: Synchronous Stream Cipher

The "simple" mode of stream ciphers works by producing a key-stream which is independent of any feedback and depends solely on the key. This means that the security of the system is completely compromised if the key is compromised. In this scheme, the sender XORs the plaintext with this key-stream to generate the ciphertext and the receiver simply XORs the ciphertext with the key-stream to obtain the plaintext. This system requires that the sender and the receiver be perfectly synchronized

The advantage of this mode is that there is no error propagation; if a ciphertext bit gets corrupted during transmission, only the corresponding plaintext bit is affected or corrupted. However, if a bit is inserted in or deleted from the ciphertext, the sender and the receiver will lose their synchronization and this will lead to corruption of the rest of the message.

In fact, the insertion attack can be used to launch attacks which can do much more harm than just simply garbling the message if the key-stream is ever reused. To understand this loophole of key reuse, let's consider a simplified example. Suppose Mike is capable of launching a known plaintext attack. What Eve does is to send P to Alice and ask Alice to encrypt the message. When Alice encrypts P with K, and sends out the corresponding ciphertext, C, Eve captures this:

$$\begin{array}{l} P = P_0, P_1, P_2 \ldots\ldots P_i \ldots\ldots P_n \\ \qquad\qquad \text{XOR} \\ \underline{K = K_0, K_1, K_2 \ldots\ldots K_i \ldots\ldots K_n} \\ C = C_0, C_1, C_2 \ldots\ldots C_i \ldots\ldots C_n \quad \leftarrow \text{Eve captures this.} \end{array}$$

Eve then inserts P_a into P and again asks Alice to encrypt it. Suppose Alice has a key-stream generator with a period of n. So, Alice encrypts this new message again with the same key-stream:

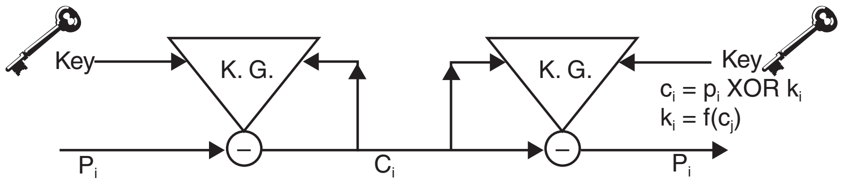
$$\begin{array}{l} P = P_a, P_0, P_1, P_2 \ldots\ldots P_i \ldots\ldots P_n \\ \qquad\qquad \text{XOR} \\ \underline{K = K_0, K_1, K_2 \ldots\ldots K_i \ldots\ldots K_n} \\ C = C'_0, C'_1, C'_2 \ldots\ldots C'_i \ldots\ldots C'_n \quad \leftarrow \text{Eve sees this.} \end{array}$$

Now, Eve knows C and C'. So she does this:

P_a XOR $C'_0 = K_0$; K_0 XOR $C_0 = P_0$; P_0 XOR $C'_1 = K_1$ and so on.

As can be seen, Eve now has the key-stream. It is for this reason that stream ciphers should never reuse the key-stream and should have a sufficiently long period.

Self-Synchronizing Stream Ciphers:



$c_i = p_i$ XOR k_i *and* $k_i = f(c_i)$

Figure 1.16: Self-Synchronizing Stream Cipher

In this mode, the state of the key-stream generator (and therefore its output) is a function of a fixed number (say n) of previous ciphertext bits. This means that the two stream generators can self synchronize with each other initially without using any external means by exchanging an n-bit Initialization Vector (IV). Most ciphers start by sending IV at the beginning of a session or message so that both the sender and the receiver can start off synchronized.

Self-synchronizing stream ciphers are better than synchronous stream ciphers in terms of security since the attacker must have access to a complete session (or at least a set of n bits) in order to decrypt any cipher bit. However, self-synchronizing stream ciphers are slower than synchronous stream ciphers since the key-stream cannot be precomputed. Also, in self-synchronizing stream ciphers an error in one ciphertext bit corrupts n plaintext bits at the receiver.

1.6 How Secure is Cryptography?

A cryptographic system has two basic ingredients: the algorithm and the keys. Since any system is only as strong as the weakest link, system designers need to keep in mind that both these ingredients need to be strong. The strength of a cryptographic algorithm is based on how it encrypts and on how it is used (its mode). The strength of a key depends on the length of the key, how long a key is used, how the key is generated, how keys are exchanged, how keys are stored, and so on.

1.6.1 Strength of a Cipher

Designing secure cryptographic algorithms is not easy and most practical implementations of cryptographic systems leave this job to a large body of academic research and use standardized algorithms. A standardized algorithm is one that has been reviewed thoroughly by a "trusted" body of academic research and "certified" to be safe. The system designers can then trust the algorithm as much as they trust the certifying body.

It is not a good idea to use a "secret" (proprietary) algorithm to ensure the security of a system since these algorithms, by definition, have been reviewed only by a small group of people and have a higher probability of hidden weaknesses and loopholes. Standardized algorithms have been reviewed much more thoroughly and therefore have a lower probability of suffering from weaknesses and loopholes. Using standardized algorithms also ensures interoperability with products from other manufacturers. Keep in mind that the fact that the inner working of the cipher is a matter of public knowledge is not as bad as it sounds since the security of the system lies in the secrecy of the keys and not the secrecy of the cipher itself. A "strong" algorithm should have the following characteristics:

- Patterns in plaintext should be concealed. Plaintext has linguistic patterns. Examples of this are—some letters occur more frequently than other; some pairs of letters occur very frequently and some letters occur only in certain pairs. It is these patterns which compromise the security of the system. A good cipher hides these patterns.
- A good mode ensures that the input to the cipher is randomized. A cipher is, after all, a mathematical operation (though a very complex one). Given the

same input, it will always produce the same output. This compromises the security of the system too, since an eavesdropper may be able to produce a mapping of most plaintext to ciphertext over a period of time.

- Manipulation of the plaintext by introducing errors in the ciphertext should be difficult. This is important to avoid attackers who have no special motive but just like to play with the system (also known as virus launchers, denial of service attacks,and so on.)
- It should be possible to encrypt more than one message with one key.

1.6.2 Key-Length: How Long is Secure?

"Security Isn't"
—Anonymous

Given a cipher, the key size is a measure of the strength of the cipher. If the key size is too small, known plaintext attacks become easy. Suppose we know a given plaintext block and the corresponding ciphertext and suppose that the key size is three bits. Given this information, we can try encrypting the input block with all the possible eight keys and see when we get the ciphertext, hence determining the key being used.

So, what is a safe key length for a cryptographic system? The answer to that question lies in the value (expressed in monetary terms) of the data the system is trying to protect. The aim of the cryptographic designer should be to make a successful attack on the system more costly than the actual worth of the data, thus making the attack unattractive for the cryptanalyst. The system designer should also keep in mind that, thanks to Moore's Law, the cost of computing power consistently decreases with time. This means that the definition of a "safe key length" changes with time.

It is extremely important to realize that using key length as a measure of the strength of a cryptographic system assumes that a brute force attack is the best possible attack against a cryptographic system.

In block ciphers, besides the key size, the size of the block (among other things) is a measure of the strength of the cipher. To realize this, assume that we have a 8-bit input block. This means that no matter how big the key and no matter how good the encrypting algorithm, there are just 256 possible output blocks that can result from the encryption of the input block. If an attacker can create a mapping between all possible input blocks and all possible output blocks then he has broken the cipher.

1.7 Beyond Cryptography

1.7.1 Firewalls

Though originally designed to contain bad network software problems, firewalls have become an integral part of network security today. Ideally, every node which connects to the network should be responsible for its own security. In most cases, however, it is not only easier but also more secure to isolate your Local Area Network (LAN) from the Internet. A firewall is basically a set of hardware and/or software that isolates a LAN from the rest of the Internet. Firewalls exist because nodes are usually unwilling or incapable of protecting themselves.

Nodes unwilling to protect themselves may sound surprising but the fact is that most PCs and UNIX Workstations are sold with a whole bunch of unsecure applications. The insecurity in these applications can be attributed either to bugs (bad programming) or to features (overriding the underlying security mechanisms to make applications more "convenient" for the user). Nodes incapable of protecting themselves include nodes like web servers which handle very high traffic loads and cannot spare the processing power required for security needs.

It is interesting to study the co-existence of web servers and firewalls. Web servers are large complex pieces of software capable of handling very high traffic load. Firewalls, on the other hand, tend to be small and simple (to avoid bugs in the firewalls themselves) but are also capable of handling very high traffic loads. Also interesting is the placement of a web server in a LAN with respect to the firewalls, keeping in mind that web servers are the easiest targets for attacks because of their sheer size and complexity. Placing a web server outside the firewall is definitely a bad idea, since it exposes the web server to all sorts of attacks. However, placing the web server inside the firewall means that if the web server is compromised, it can then be used as a launching pad for attacks on other hosts in the LAN. The typical solution for this problem involves the use of two firewalls. The web server is then placed in the so called demilitarized zone (the region between the internal firewall and the external firewall).

Though firewalls are by no means sufficient protection against security threats, they form a good first line of defense against attacks coming from outside the LAN. Firewalls have also come a long way in terms of functionality and versatility. Even though the underlying theory of firewalls continues to be access control (keep out the bad packets), there is no doubt that firewalls have evolved over the years. Firewalls today can operate at the link layer (packet filtering based on MAC addresses), the network layer (packet filtering based on IP addresses) or the application layer (packet filtering based on the data content inside the packet).

It is important to realize that firewalls implement user-defined policies and the security of a firewall is a factor of the strength of the policy. Stricter policies mean tighter access control and therefore more security, but this comes at the cost of limiting the applications that users might wish to use. The job of the network administrator, therefore, is to strike a balance between these two needs.

1.7.2 Denial of Service Attacks

Cryptography also does not provide any help against Denial (or Degradation) of Service (DoS) attacks. To understand why this is so, we need to understand what DoS attacks are and how they are launched.

DoS attacks work by overloading the finite resources available in a system, thus making the system unusable by users. They are sometimes categorized based on the resource they overload; for example, attacks may overload bandwidth in the access loop, processing capacity at the routers, processing capacity at the server, or any combination of these. A trivial example is a Java™ applet running in an infinite loop, forking off threads which will sooner or later exhaust CPU and memory resources in the system. A more practical example (and one which has been used to launch attacks in the recent past) involves a TCP client application sending SYN packets to a server in a loop. Since the TCP at the server allocates resources and keeps track of half-open connections in a table, if the client sent enough SYN packets to exhaust this table, the server would reject all future connection requests, thus denying service to other clients.

Yet another variation of such attacks is the Distributed Denial of Service (DDoS) attack where attackers uses common exploits to install a zombie program on as many machines as they can. The zombie program binds itself to a port and waits. The zombies can come alive at a predetermined time or on receiving an explicit message from the attacker. On waking up, the zombies send a large number of packets to the desired target, thus trying to exhaust resources at the target. DDoS attacks are much more effective than the DoS attacks since the target has to cope with much higher loads. Also, since packets are coming from multiple clients and therefore multiple IP addresses, it makes these attacks harder to track down the attacker.

Defenses against DoS attacks may be based on protection, reaction or prevention. Protection works by having multiple resources where one is needed. This means that if one resource is brought down by the attack, the other resource can take over without the service being affected. Reaction works by tracking the source of the packets which are overloading the resource and then taking appropriate action against the source of the attack (usually this means shutting off the source). However, reaction will not work if the attack is using IP address spoofing or if it is a DDoS attack. Prevention of DoS attacks involves the use of ingress filtering wherein each Internet Service Provider (ISP) ensures that the packets coming from the endpoint into the network are not using address spoofing. If each ISP uses this prevention mechanism, it would make it easier to track the attack source (and therefore be a strong deterrent to attackers) in case of an attack. However, there is no current law requiring ISPs to do this and therefore this prevention scheme is still largely theoretical.

1.7.3 Code Security

Finally, cryptography (or for that matter any other branch of science) cannot protect against badly written code. In this section, we briefly see some of the most common code vulnerabilities which have been exploited in the recent past.

Buffer Overflows: These attacks exploit vulnerabilities in applications, programming languages and operating systems. The underlying problem is that some applications do not do argument bound checking. This means that there is no check in the application code for ensuring that the length of the supplied argument is within certain bounds. Since some programming languages do inherent argument bound checking and others don't, this can either be seen as a vulnerability in the application code or a vulnerability in the programming language, depending on which side of the fence you sit. As we shall see, the operating system vulnerability which makes such attacks even more dangerous is the fact that most operating systems allow code to be executed from the stack (for example, the value of the function pointer of the calling routine is stored in the stack).

To understand buffer overflows more clearly, consider the old version of the UNIX application sendmail. This application could be invoked in debug mode using two arguments. The first argument was an index into an array and the second argument was the value to be written into that array element. The bug was that the application did not do any bound checking on either of the arguments. This meant that users could, in fact, write anything (using the second argument) to any memory location (using the first argument) by manipulating the arguments correctly. Not only this, users could in fact change the course of program execution and execute their own piece of code (worms, and so on) by first placing this code in a certain memory location and then using sendmail to overwrite the return function pointer to jump to their own memory location.

The best (and often the simplest) protection against buffer overflow attacks is to ensure that your application does not have such bugs. This can be done by using tools like lint and dynamic code analysis tools.

TOCTTOU: These attacks exploit the fact that there is a finite time from time-of-check to time-of-use. An example would make things clearer. UNIX stores user passwords (or rather their hashes with salt) in /etc/pwd file. When users want to update their passwords, UNIX does the following:

1. Pick a random filename, say, rand123.txt
2. Check if rand123.txt exists in /tmp. If yes, goto 1
3. If not, open /tmp/rand123.txt.
4. Copy contents of /etc/pwd to rand123.txt
5. Add new entry.
6. Copy rand123.txt to /etc/pwd.

The vulnerability that TOCTTOU exploits is that the system call required for step 2 (checking) is separate from the system call required for step 3 (opening) and therefore there is a finite time between these two steps. To understand how the attack exploits this, let us assume that the attacker can guess the random filename the system generates (this is not as tough as it sounds since the "random" filename uses the current timestamp as a seed for the random number generator). Now, what a hacker needs to do is create a fake password file with this random name and copy this file into the /tmp directory between steps 2 and 3. If the attacker can do this, they have successfully replaced the real password file with their own version of the password file.

1.7.4 Steganography

As we have seen, cryptography is used to satisfy most of the requirements of secure communications today. However, there are certain aspects of security that cryptography can't protect against; for example, cryptography can "hide" the information content of the message but it cannot hide the fact that a message was in fact sent. To achieve this, we need to use steganography. Steganography involves embedding a message inside another message; for example, we may hide a text message within another text message such that the hidden message can only be revealed by XORing the sent message with a secret bitmap. Recent uses of steganography involve hiding messages inside a JPEG image by exploiting the fact that in JPEG, changing the lower bits in a pixel representation does not change the appearance of the image enough to be detectable by human eyes.

1.8 Conclusion

In this chapter, we saw what network security means and how cryptography forms the basis of the protection mechanisms we can use to provide this security. The next chapter will look at how cryptographic protocols are used in networks today to provide security.

CHAPTER 2

Network Security Protocols

2.1 Introduction

In Chapter 1, we looked at how cryptography forms the basis of network security. The three important aspects of network security are authentication, encryption and message integrity. Cryptography provides a mechanism to achieve each of these objectives.

The concept of "keys" is central to the idea of cryptography. There are two important concepts related to keys: key generation and key distribution. The former refers to how keys are generated and created, whereas the latter refers to how a user node in a network finds out the key(s) used to talk securely to other users or nodes. The term *key establishment* is a little vague since its exact meaning depends on the context in which it is used. We look at key-related network security protocols in Section 2.2. Then in Section 2.3, we look at network authentication approaches and protocols. The approach taken in both these sections is to study not only the standardized protocols in use today but to also look at a build-up to these protocols to understand the intricacies in authentication and key-establishment process. Sections 2.4 and 2.5 study some of the most common encryption and integrity protocols in use today.

2.2 Key Establishment Protocols

For Symmetric Key Cryptography (SKC), the term *key establishment* can be clearly broken down into two components: Key generation and key distribution. For Public Key Cryptography (PKC), however, the term key establishment is a little more complicated. In the case of two users communicating with each other over an unsecure channel, the use of PKC key establishment protocols results in both users *establishing* the public-private key pair. Specifically, when the PKC establishment protocol ends, each user has a private key (known only to themselves) which has never been transmitted onto the medium and a public key (known to the other end as well) which, even if available to an eavesdropper does not compromise the security of the system. However, as we will see in Section 2.2.3, this approach is susceptible to "man-in-the-middle" attacks. To protect against such attacks, we have to involve another layer of key distribution in PKC. Don't worry if it's a little confusing right now. Things will become much clearer as we move on.

2.2.1 Key Generation in SKC

In SKC, key generation is relatively simple. The only requirement of a SKC key is that it be random[1] and long enough to deter a brute force attack. Hence any random string or number can be used as a key, as long as both communicating users and only both communicating users know the key.

2.2.2 Key Distribution in SKC

For understanding the problem of key distribution, recall that in SKC network security is achieved by ensuring that the communicating users and only the communicating users know the shared secret (key). Now, consider a network with n nodes where each node may wish to talk securely to every other node. How many keys would this require? If you are math lovers, you will immediately realize that the answer is ${}^{n}C_{2}$. For the rest of us, see the table below to appreciate the scale of key growth:

Number of users in the Network	Number of Keys Required (SKC)
2	1
3	3
4	6
.	.
.	.
50	1,225
.	.
.	.
250	31,125
.	.
.	.
N	{N * (N-1)}/2

Since the number of keys required grows as O (n^2), where n is the number of users in the network, we have a scaling problem. A growth in the number of keys which need to be kept secret makes administration more and more difficult. Why? Because each node in the network needs to be configured to store the keys of all the other nodes it wishes to communicate with. How do we configure so many keys at each node? One approach is to require manual configuration of all keys into each node in the network. However, this approach does not scale well either. Also, it is highly susceptible

[1] Pseudorandom, to be precise.

to human error. What is needed is a scaleable process where human intervention is minimized. In other words, we want the network to be able to obtain keys without requiring human intervention. But how do we distribute the key securely if the network is not secure to start communicating in the first place?

This is a typical chicken and egg situation[2] which is characteristic of SKC. SKC relies on a preshared secret between the communication parties to secure communication between them. It does not say how to establish the "preshared" secret. Undoubtedly, this is a tough problem, since we cannot trust the network.[3]

To solve these two problems, most SKC implementations use a Key Distribution Center (KDC). The KDC is a centralized trusted third party which stores keys for all the n nodes in the network. In other words, each node in the system is configured with only one key—its own. However, this key is also stored at the KDC. This makes key administration much easier.[4]

Now, consider a network of n nodes where Alice wants to talk to Bob securely. To do this, Alice first establishes a secure session with KDC using her own key[5] and requests the KDC to generate a session key for her session with Bob. On getting this message, KDC establishes a secure session with Bob (using Bob's key) and then sends the same session key to both Alice and Bob. Alice and Bob now use this session key to establish a secure session. The obvious drawback of the scheme is that centralization of all information at the KDC creates a performance bottleneck and a single point of failure or security compromise.

2.2.3 Key Establishment in PKC

Recall that for two parties to communicate securely using PKC, we need two keys instead of the single key required in SKC. The two keys are comprised of the public key (available to anyone) and the private key (known only to the user). The security of the system lies in keeping the private key secret. One of the basic requirements for the "public-key, private-key" pair in PKC is that the keys be complementary; in other words a message encrypted with one of the keys should be decryptable only with the corresponding key in the pair. We look at key generation protocols for PKC in the next few sections.

[2] Or an egg and a chicken situation? ☺
[3] …otherwise we wouldn't be using cryptography to secure our communication.
[4] As a rule of thumb, simpler systems are more secure since they reduce the probability of error (human and otherwise).
[5] This is possible since Alice knows her own key and the KDC knows the keys of all users in the network.

In this section, we look at key distribution protocols for PKC. On the face of it, this might seem like a trivial problem. Once we have established the "public-key, private-key" pair securely over an unsecure medium, key distribution can be an "ask-before-use" algorithm.[6] Such an approach would require that when Bob wants to start communicating with Alice, Bob simply asks Alice for her public key and Alice sends her public key to Bob. Another approach could be to have each user broadcast her public key, which makes her public key available to all nodes in the network. Since making the public key "public" does not result in a security compromise,[7] this approach should not result in compromising the security of the system. Unfortunately, this is not true. Section 2.2.4 shows why this is so, and Section 2.2.5 shows how to solve the problem using a trusted third party.

The trusted third party in PKC is the Certificate Authority (CA), whose role is best understood by an example. Suppose Bob wants to talk to Alice using PKC. To find out Alice's public key, Bob should ask the CA for Alice's public key. When the CA receives Bob's request for Alice's public key, the CA responds back with a certificate. This certificate is basically of the form EK_{iCA}\{Alice's public key is K_{wa}\}. In other words, the CA sends the message "Alice's public key is K_{wa}" encrypted with its own private key. When Bob receives this message, it uses the CA's public key (K_{wCA}) to "decrypt" the certificate and obtain Alice's public key. Since the CA's private key is known only to the CA, no one can forge the certificate and claim another key as Alice's public key.

Using the certificate authority for key distribution in PKC means that Bob can get any other user's (node's) public key securely[8] as long as he has the CA's public key and as long as he trusts the CA. This means that the man-in-the-middle attack is no longer a threat to the system, but this also means that each node in the system already *knows* the public key of the CA. This can be achieved by configuring each user with the public key of the CA, or by publishing the CA's public key in a "well-known" medium. Note that publishing one public key is much easier than having each user publish her public key.

Even though CAs suffer from some of the same drawbacks as a KDC (single point of failure and compromise), they offer some advantages over KDC. First, the CA is not a performance bottleneck since it is not actively involved in user sessions. Second, the

[6] Similar to the SKC-based key distribution scheme.
[7] One of the basic tenets of PKC is that the security of the system should lie in the private key and not the public key.
[8] And thus avoid a man-in-the-middle attack.

information stored at the CA (the certificates) is not sensitive in that even if the CA is compromised, the security of the system is not compromised.[9]

2.2.4 Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange algorithm is based on the ease of doing exponentiation and the difficulty of doing discrete logarithms (the reverse operation). Even though it cannot be used for encrypting messages or for signing messages, it allows Alice and Bob to establish a shared secret key securely over an unsecure medium. Figure 2.1 shows how the Diffie-Hellman algorithm works.

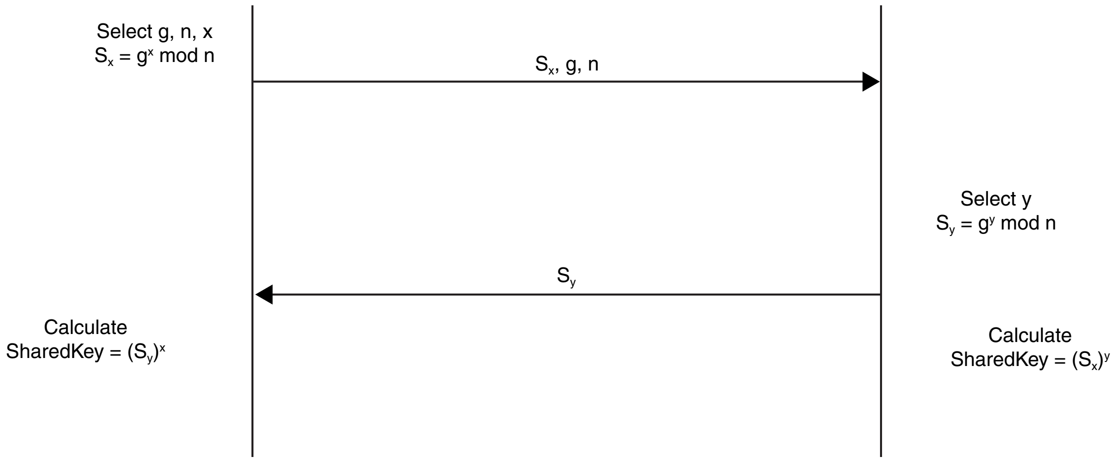


Figure 2.1: Diffie-Hellman

The protocol is beautiful in its simplicity. It consists of only two message exchanges. First, Alice selects a large prime number n, a generator g and a random number x and calculates $S_x = g^x \bmod n$. As the first message, Alice sends g, n and S_x to Bob. When Bob receives this message, he generates a random number y and calculates $S_y = g^y \bmod n$ and sends it to Alice as the second message. At this time, both Alice and Bob can calculate $S_{xy} = g^{xy} \bmod n$ easily. Alice can calculate $(S_y)^x \bmod n$ and Bob can calculate $(S_x)^y \bmod n$. Both evaluate to the same number $g^{xy} \bmod n$ (called S_{xy}) and this is the shared secret.

The only secret numbers in the protocol are x and y. Only Alice knows x and only Bob knows y. No effort is made to keep g, n, S_x or S_y secret. Let's suppose Eve captured the

[9] The system is secure as long as the attack does not compromise the CA's private key allowing Eve to issue invalid certificates.

two messages exchanged between Alice and Bob. This means she has access to g, n, S_x (g^x mod n) and S_y (g^y mod n). However, Eve still cannot calculate S_{xy} (g^{xy} mod n) since it is hard to find x given S_x (g^x mod n) or to find y given S_y(g^y mod n) using discrete logs.

Even though the Diffie-Hellman algorithm is beautiful in its simplicity, it has its own loopholes. One major loophole is that it is susceptible to a man-in-the-middle attack as shown in Figure 2.2.

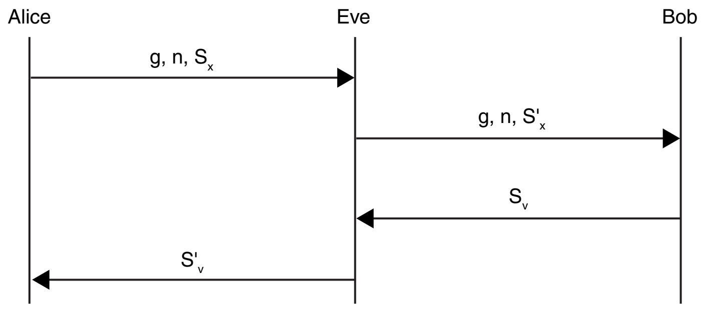


Figure 2.2: Man-in-the Middle Attack Against Diffie-Hellman

In this attack, Eve captures the first message that Alice sends to Bob, notes down g, n, S_x and modifies the message being sent to Bob so as to replace S_x (g^x mod n) with S'$_x$ (g^k mod n) where k is a secret number known only to Eve. Similarly, when Bob sends the second message containing S_y to Alice, Eve replaces S_y (g^y mod n) with S'_y (g^pmod n) where p is a secret number known only to Eve.

At this point, Alice and Eve have a shared secret of (g^{xp} mod n) and Bob and Eve have a shared secret of (g^{ky} mod n). The only problem is Alice and Bob think they are talking to each other whereas in reality they are both talking to Eve. This is a dangerous attack, since Eve may now act as a relay between Alice and Bob, thus acting as a transparent eavesdropper between them. He may actually hijack the two sessions completely, carrying out independent conversations with Alice and Bob where neither Alice nor Bob is aware that the session has been hijacked.

The man-in-the-middle attack works against the Diffie-Hellman (DH) protocol since there is no inherent authentication mechanism in the protocol and therefore no way for Alice and Bob to verify that they are in fact talking to each other. To solve this problem is the topic of the next section.

2.2.5 Enhanced Diffie-Hellman Key Exchange

The man-in-the-middle attack loophole in DH protocol stems from the fact that Alice has to inform Bob of S_x over an unsecure medium and Bob has to inform Alice of S_y over an unsecure medium. Additionally, since g and n are also transmitted over the unsecure medium, it is trivial for Eve to generate a fake S_x and a S_y.

To protect DH against man-in-the middle attacks, we exploit the Public Key Infrastructure (PKI) described in Section 2.2.3. The PKI relies on a single trusted party—the certificate authority—which signs the public key of a user with its own private key to create the user's certificate.

For the Diffie-Hellman key exchange, if g and n are fixed it is known as static Diffie-Hellman and the CA creates a certificate of the form $EK_{iCA}\{g^x \bmod n, \text{Alice}\}$for Alice and $EK_{iCA}\{g^y \bmod n, \text{Bob}\}$for Bob. On the other hand, if g and n are ephemeral (to be established dynamically), it is dynamic Diffie-Hellman and the CA creates a certificate of the form $EK_{iCA}\{g, n, g^x \bmod n, \text{Alice}\}$for Alice and $EK_{iCA}\{g, n, g^y \bmod n, \text{Bob}\}$for Bob.

Now, Alice knows her private key x, gets Bob's certificate, derives Bob's public key S_y (= g^y mod n) from Bob's certificate and calculates the shared secret as $(S_y)^x$. Similarly Bob calculates the shared secret as $(S_x)^y$.

In this new scheme, the vulnerability against man-in-the-middle attack is no longer present since Eve (even if he knows g and n) can no longer fool Bob into believing that A's public key is (g^k mod n) and not (g^x mod n) since public keys must now be signed by the CA's private key.

2.2.6 RSA

The DH public key algorithm described above is the oldest key establishment protocol still in use today. Another important (and more recent) key establishment protocol is the Rivest-Shamir-Adleman (RSA) protocol, which is based on the difficulty of factoring the product of two large primes. However, RSA is a much more generic (powerful) protocol than DH in that it can be used for encryption and integrity (digital signatures) also.

To put things in perspective, realize that the DH protocol went far enough to establish a shared secret securely between two parties communicating over an unsecure medium. Though this is undoubtedly a great achievement, the DH protocol says nothing about how to use this shared secret for encryption or integrity.

In RSA, to generate a private-key, public-key pair, Alice selects two large prime numbers p and q and calculates n as the product of p and q. Alice also calculates phi(n)[10] as (p–1) * (q–1). Next, to generate the public key, Alice chooses a number e such that e is relatively prime to phi(n). Then, the public key is defined as the pair <e, n> and the private key is defined as the pair <d, n> where d is the multiplicative inverse of e mod phi(n).

The process described so far may sound trivial to implement but is really not. The security of the system lies in the prime numbers p and q; the larger these primes, the more secure the system. Commercial implementations typically use 256-bit p and q. However, it is not easy to generate such large primes. In fact, it is so difficult to find large prime numbers that most implementations of RSA use probable-primes, which are defined as numbers which are "probably prime." These numbers are typically generated using Fermat's Little Theorem.[11]

Figure 2.3 shows how RSA is used for encryption. When Alice wants to protect the message that Bob is sending to her, she calculates RSA keys as described above and sends her public key to Bob. Bob breaks up the message he wants to send to Alice into blocks. The size of each block must be less than the size of the key length.[12] Bob then encrypts each plaintext block (let's call it m) to obtain the ciphertext block (let's

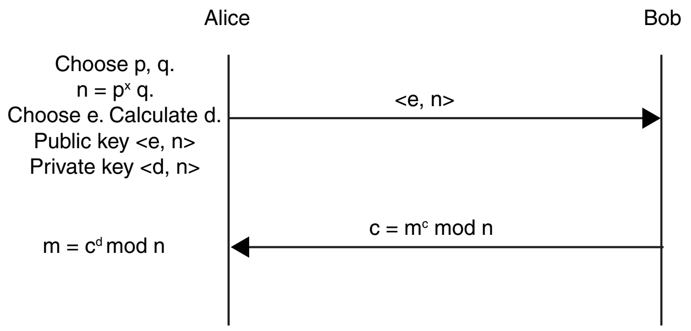


Figure 2.3: RSA

[10] In modular arithmetic, phi(k) is defined as the number of elements less than k which are relatively prime to k modulo k. For a prime k, it can be shown that phi(k) = k–1. Also, it can be shown that phi(a *b) = phi(a) * phi(b) = (a–1) * (b–1).

[11] Fermat's little theorem states that if p is a prime number, and x is a number such that 0 < x < p, then x^{p-1} = 1 mod p. Even though the theorem may hold true for some nonprime p's too, this is a rare occurrence and therefore a probabilistic primality test for a number n is to pick a number a less than n and compute an-1 mod p. If the answer is not 1, n is definitely not prime. If the answer is 1, n is probably prime. Obviously, the more numbers are tried in place of a, the higher the probability that n is prime.

[12] The key length is variable and depends on how large p and q are. For p and q 256 bits long, the key would be around 512 bits.

call it c) as c = m^e mod n. When Alice receives the ciphertext, she obtains the plaintext as m = c^d mod n. This works because e and d are multiplicative inverse of each other mod phi(n). Therefore, {c^d mod n} = {$(m^e \bmod n)^d$ mod n} = {m^{ed} mod n} = {m}.

Why can't Eve decrypt the ciphertext even though she knows e and n? Because it is not easy to calculate d without knowing phi (n). Note that phi(n) is trivial to calculate if p and q are known but is very difficult to calculate if p and q are not known. Therein lies the security of RSA.

Figure 2.4 shows how RSA is used for message integrity using digital signatures. The aim here is to assure Bob that the message he received from Alice has not been modified by anyone in transit. Note that we are not interested in confidentiality or authentication of the session here: rather only in the integrity of the message.

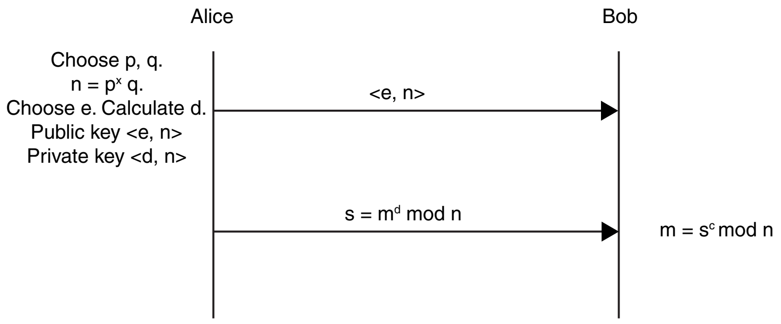


Figure 2.4: Using RSA for Message Integrity

To achieve this, Alice calculates RSA keys as described above and sends her public key to Bob. Now Alice breaks up the message she wants to send to Bob into blocks. As before, the size of each block must be less than the size of the key length. Alice then signs each plaintext block (let's call it m) to obtain the signature (let's call it s) as s = m^d mod n. Now, Alice sends s to Bob. When Bob receives s, he obtains the plaintext as m = s^e mod n. Again, this works because e and d are multiplicative inverse of each other mod phi(n). Therefore, {s^e mod n} = {$(m^d \bmod n)^e$ mod n} = {m^{ed} mod n} = {m}.

Note that even Eve can obtain m from s, since she can eavesdrop on e and n but confidentiality is not what we are trying to achieve here. The aim here was to assure Bob that the message was sent by Alice. This scheme works because if Bob can obtain m from s using Alice's public key <e, n>, he can be assured that the message was signed using Alice's private key <d, n> which only Alice knows.

As with the DH algorithm, the RSA algorithm too would suffer from the man-in-the-middle attack if used per se. To circumvent this loophole, the RSA algorithm is used with the PKI where the public keys of all communicating parties are stored at the central trusted certificate authority as explained in Section 2.2.

2.3 Authentication Protocols

The term authentication refers to the process of verifying that a node or user is who they claim to be. One of the primary uses of authentication in networking is to implement access control reliably. Controlling access to the network is one of the primary defense mechanisms in network security. If you allow only people you trust to access the network, you have an inherent protection built into the system. Authentication implements access control by securely identifying users or nodes trying to access the network.

2.3.1 Address-Based Authentication

For any network to be able to route messages, calls or packets, it is imperative that each node in the network be assigned a unique address. The format of this address depends on the routing protocol being used in the network. Given that each node has an address, a first-step approach to controlling access into the network is to allow only a predetermined set of addresses to access the network. In other words, this approach authenticates the user or node on its address.

At this point, it is important to distinguish between a user and a node. A user is a human, whereas a node is the machine the user is using to connect to the network. The address used for this approach usually refers to the address of the node. In IP networks, for example, this address may be the message authentication code (MAC) address[13] or the IP address. So, an address-based authentication scheme in an IP network may be implemented by the switch (or router) allowing only a preconfigured set of MAC or IP addresses to access the network.

What are the loopholes in this scheme? The loopholes are based on the fact that the assumptions that the scheme makes are easy to violate. First, it assumes that there is a strict one-to-one relationship between a user and a node. This may not be true where a single node is used by multiple users or where a single user can use multiple nodes. This means that authenticating a node based on its MAC address does not really authenticate the user.

[13] The MAC address of a device is a 48-bit number assigned by each manufacturer. Each device in the world is supposed to have a unique MAC address.

Second, this scheme assumes that the MAC address of a node is fixed and cannot be altered or spoofed. Unfortunately, this is a bad assumption since it is usually pretty easy for Eve to transmit a packet claiming any MAC or IP address as its address. Spoofing a MAC address may be a little more difficult since some network access devices now have a built in mechanism to protect against such alterations, but most do not. Spoofing an IP address on the other hand is almost trivial since IP-address discovery uses address resolution protocol (ARP), which works on an inherent assumption of trust on all nodes in the local area network (LAN).

Even though address-based authentication is not sufficient by itself, it forms a good additional obstacle or deterrent when used along with other access control mechanisms like authentication protocols.

2.3.2 Passwords for Local Authentication (Login)

Passwords are probably the oldest and the most common way of providing authentication. They are used for logging into either local machines or into remote machines. Even though we are dealing with the latter, it is instructive to see how the former works.

A first attempt at using passwords would involve storing the <username, password> pair list in a file on the machine. When a user tries to login, they are prompted for their password and the value they provide is compared with the corresponding value stored in the file. The permissions for the database file containing the list of <username, password> pairs can be set so that it is accessible only to the administrator.

What's the problem? Consider what happens if the password file is compromised. All user passwords are compromised. In other words, the security of the whole system relies on a single (weak) link: the password file. This is not a good security architecture. Also, some users may not want to trust the administrator with their passwords. (Would you want to trust your Internet service provider (ISP) with your login password?)

To solve the above issues, we can modify the scheme to have the machine store <username, hash (password)> pairs. When a user tries to login using their password, the node calculates the hash of the password and compares this value with the stored value. Since the password file only stores the hash of the passwords and since hash functions are one-way functions, even if the file is compromised the passwords are still secure. This is a great improvement from the previous scheme, but it's still not foolproof. This scheme is open to something called a dictionary attack.

The dictionary attack exploits the fact that humans tend to use dictionary words as passwords; in other words most passwords are words which can be found in the dictionary. Based on this assumption, Eve may compute the hash of all words from the dictionary and store this <word, hash (word)> pair list. Then, once she gets hold of the password file on a machine, Eve searches the hash (password) in his stored list and obtains the corresponding word as the password.

The dictionary attack belongs to a group of attacks known as offline attacks. Such attacks are easier to mount since they require most of the processing-intensive work to be done before the attack is actually launched. In the case of dictionary attacks, since most of the computation required in the attack (calculating the hash of all words in the dictionary) can be done without involving the password database, it is an offline password guessing attack. To protect against dictionary attacks, we can modify our protocol to store the <username, hash (salt + password), salt> triplet in the password file, where salt is an n-bit random number. When the user attempts to login, the node uses the provided password and the salt value from the triplet stored in the password file to calculate the hash (salt + password). It then compares it with the hash value stored in the password file and makes a decision. How does the salt help? To launch a dictionary attack against this protocol, Eve needs to calculate the hash (salt + *dictionary_word)* for each *dictionary_word:* but remember that Eve does not have access to the salt before the attack is launched. So, the only alternative Eve has is to calculate the hash (*all_possible_salts* + *dictionary_word*) for all dictionary_words. We have therefore made the job 2^n times harder for Eve, where n is the length of the salt in bits.

One could argue, correctly, that Eve could do the hash calculation for all dictionary words after it gets access to the password file (and hence the salt). However, this makes the dictionary attack an online attack which is much harder to launch. Online attacks require that the system be compromised before the bulk of the computation starts. In other words, these attacks attempt to break the system security in "real-time" and are therefore more difficult to launch. The simplest example of such attacks is the brute force attack where Eve actively tries out different passwords to login. Such attacks can be deterred (besides using cryptography) by enforcing system policy wherein an account is temporarily locked after someone has attempted to log in with a wrong password more than k times.

Now that we have seen how passwords are used for local authentication, we are ready to move on to the next section.

2.3.3 Passwords for Network Authentication

Using passwords for network authentication is a little different from using them for local login. The primary difference is that even the hashed password cannot be sent across on the network since the hashed password can be used by Eve to launch a dictionary attack offline. We solved this problem for local authentication by using salts. However, salts do not work for networks. Why? Consider our old friends Alice and Bob. Suppose Alice wants to authenticate with Bob. Using salts to achieve this would work something like shown in Figure 2.5.

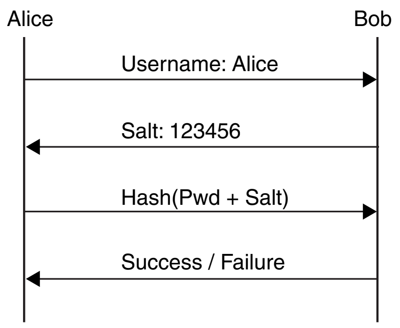


Figure 2.5: Using Passwords for Network Authentication

The basic problem with this approach is that Eve can easily get access to the salt (since it is transmitted as plaintext) by recording the first session. Eve can then go offline, do all the computation intensive work, recover Alice's password and then use it at will. In effect, the transmission of the salt in plaintext allows the dictionary attack to remain an offline attack. For this reason, passwords are used differently in network authentication.

For network authentication, passwords serve as the "seed" for deriving keys to be used in challenge response systems. So, how do we convert a password to a key? One way of doing this could be to take a hash of the password and then use a subset of the hash as the key. There are other ways of doing this too, but the basic idea is that the security of the system relies on the fact that the key can be derived only from the password and that neither the password nor its hash (the key) are ever transmitted onto the network. Theoretically, there is no need for passwords. If users can remember their key, we don't need passwords. The problem is that most humans are better with words than they are with long alphanumeric strings. It is for this reason that we use passwords in network authentication. The human user remembers a password which is converted to the key by the node (the machine) locally (without being ever transmitted onto the network).

2.3.4 Authentication Using SKC

Before we get into how SKC is used for authentication, it is important to keep in mind that SKC assumes that there exists a preshared key between the users (nodes).

2.3.4.1 One-way Authentication Using SKC

Figure 2.6 shows how SKC is used for achieving one-way authentication. In this example, Alice wants to authenticate with Bob. The process starts by Alice sending her “username” to Bob. Next, Bob sends a random number (a challenge) to Alice and stores this locally. Then, Alice encrypts the message with the shared secret key (derived from her password) and sends the result back to Bob (the response). When Bob receives the response, he encrypts the challenge he sent with the shared secret key and compares it with the received value. If both these values match, Bob can be sure that he is communicating with somebody who knows the secret key. Assuming that only Alice and Bob know the secret key, Bob has ensured that he is communicating with Alice.

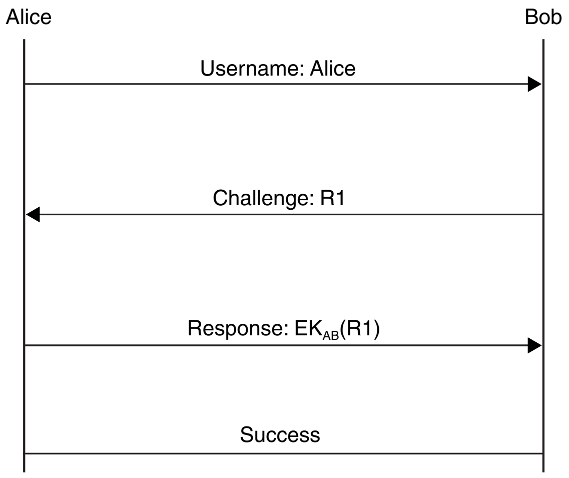


Figure 2.6: One-way Authentication Using SKC

Figure 2.7 and Figure 2.8 show variations of how SKC can be used for authentication. In Figure 2.7, Bob sends the encrypted challenge to Alice, Alice decrypts it and sends back the plaintiff’s challenge to Bob. This is just another way for Bob to verify that Alice knows the shared secret key. There is one significant difference between Figure 2.6 and 2.7. In 2.6 one way-hash functions can be used in place of encryption; however in Figure 2.7 using encryption is a must since this scheme requires that the challenge to response transition be reversible.

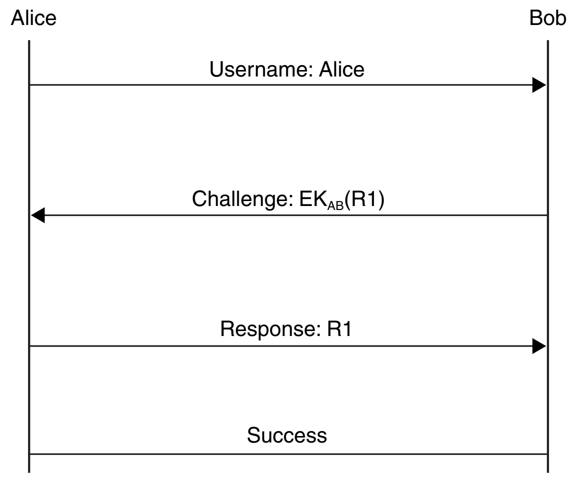


Figure 2.7: One-way Authentication Using SKC—*Variation*

Figure 2.8 reduces the number of messages required for authentication to just one message, thus making it an attractive option for systems demanding backward compatibility with "no authentication systems." Also, in this scheme Bob can be stateless. This is a big deterrent against denial of service (DoS) attacks.[14] In a sense, Figure 2.8 uses the current timestamp as the challenge and the communicating parties encrypt this challenge.

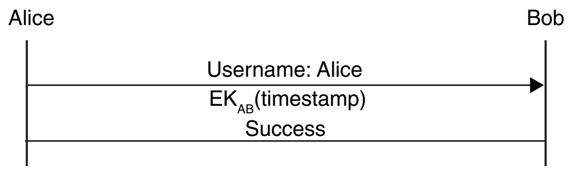


Figure 2.8: One-way Authentication Using SKC—*Variation*

However 2.8 suffers in that it has a strict requirement for Alice and Bob to be synchronized in time. Since time synchronization in the network (especially large networks) is not a trivial task, this means that the security of the system is dependent on the network time protocol being used. Obviously, this is not a desirable feature. Additionally, if the authentication protocol shown in Figure 2.8 is used with stream ciphers, it leads to a major security loop hole. In stream ciphers, flipping 1 bit in the cipher text flips 1 bit in the plain text. Therefore, Eve may use an approximate

[14] Note that DoS attacks work by overloading the memory (usually) of the target system. If the target system is stateless (does not store any information), DoS attacks are harder to launch.

timestamp and try flipping the millisecond bits to get the approximate timestamp well within the allowable skew of the network-time protocol being used.

2.3.4.2 Mutual Authentication Using SKC

Figure 2.9 shows how SKC can be used for mutual authentication. The aim here is for Alice and Bob to authenticate each other. A first attempt to achieve this would be to extend Figure 2.6 in both directions and this is exactly what Figure 2.9 does. Bob sends a challenge to Alice and expects her to encrypt[15] it correctly using the shared secret key. Correspondingly, Alice sends a (different) challenge to Bob and expects him to encrypt it correctly using the shared secret key. However, this scheme requires an exchange of five messages. This is inefficient in terms of bandwidth, time, processing power, and so on. To make the system efficient, we look at Figure 2.10 as an alternative.

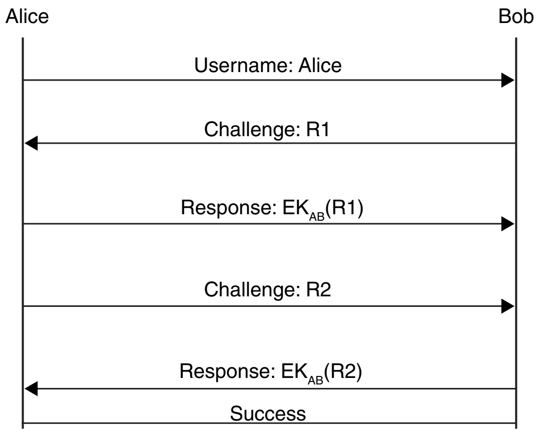


Figure 2.9: Mutual Authentication Using SKC

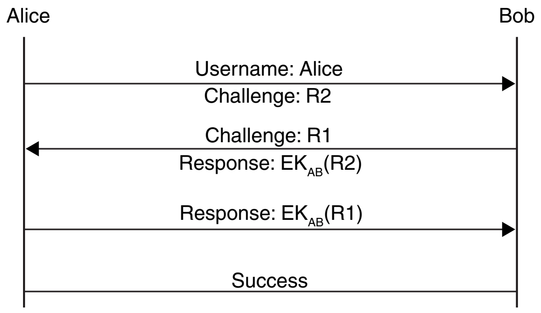


Figure 2.10: Mutual Authentication Using SKC—Reduced Messages

[15] Or hash it.

In the first message, Alice sends her challenge along with her username; in other words we combine messages 1 and 4 of Figure 2.10. On receiving this message, Bob responds back with the response and his own challenge to Alice. Finally, Alice responds back with her response to Bob's challenge. This results in the use of only three messages to achieve mutual authentication. However, this "efficiency" comes at a cost: it makes the system open to reflection attack. The reflection attack is shown in Figure 2.11.

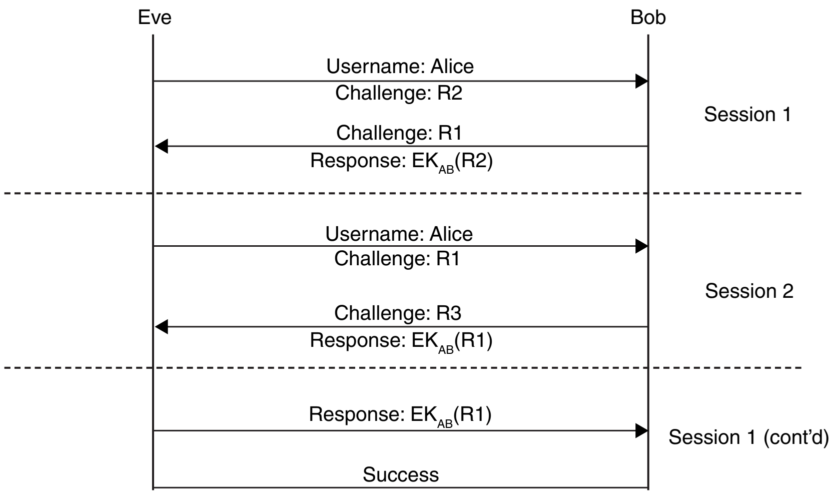


Figure 2.11: Reflection Attack

Suppose Eve wants to impersonate Alice to Bob. To achieve this, Eve sends a message claiming to be Alice, and a challenge R2. Since Bob has no way of knowing (yet) that this message came from Eve and not Alice, Bob responds with the response which is K_{AB}(R2) and another challenge R1. At this point, Eve initiates another session with Bob by sending a message to Bob claiming to be Alice and the challenge R1 it received from Bob. Again, Bob responds with K_{AB}(R1) and another challenge R3. In effect, Eve exploits the algorithm to have Bob produce a response to his own challenge.

The reflection attack can occur if Bob allows multiple sessions from the same user or if Alice uses the same key at multiple servers.[16] On the other hand, prevention against reflection attack may be ensured by enforcing system-wide policies. In client-server

[16] If Alice uses the same password at multiple servers, it does not necessarily mean that Alice uses the same key at these servers since the algorithm which converts the password into the key may take the server name or identity into account for creating the key.

architectures for example, a policy which would protect against reflection attack would be "Challenges from clients to servers should follow a unique format and challenges from servers to clients will follow another unique format."[17] Another policy can be to use two secret keys instead of one secret key—one key for each direction of authentication.

2.3.4.3 Lamport's Hash

Note that all authentication schemes described until now used SKC and required Bob to store the <username, SharedKeyForThisUserPassword> for each user. This means that if this database was compromised, the attacker can easily impersonate Alice. PKC-based authentication schemes do not suffer from this limitation as we see in the next section. However, PKC is also computation intensive. This is where Lamport's Hash comes in.

Lamport's Hash is an authentication protocol which uses SKC but still protects against the case where Bob's database is compromised. Note that in all SKC authentication schemes explained until now, Bob needed to store the <username, password> pair for Alice. To implement Lamport's Hash, Bob has to store[18] the <username, $\text{hash}^{\text{raise}}$ (password)> triplet for Alice. The variable "raise" is an integer which is initialized to a large number, say 1000.

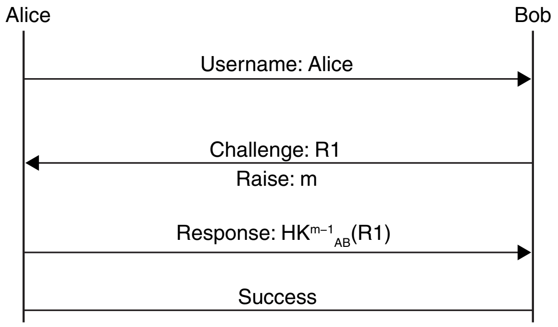


Figure 2.11a: Lamport's Hash

When Alice wants to communicate with Bob, she sends her username to Bob. Bob sends back a challenge R1 and the current value of "raise," say m. Alice then computes the hash of the challenge R1, (m–1) times; in other words Alice calculates $HK_{AB}{}^{m-1}(R1)$ and sends this as the response to Bob. Bob takes the received quantity, hashes

[17] For example, client to server challenges would be odd and server to client challenges would be even.
[18] Or the KDC had to store.

it once and compares it with the stored value. If the values match, Bob sends authentication success to Alice, replaces the stored quantity with the received quantity and reduces the value of the raise by one to m–1.

When the value of the "raise" at Bob reaches 1, the user needs to change her password and Bob needs to be reconfigured securely with the new password—which must be done by the network administrator. An enhancement to Lamport's Hash is to use a salt value along with the password before taking the hash. This allows the user to keep using the same password even when n reaches 1, since n can be reset to 1000 with a new salt. The salt also helps by increasing the life of a password by having each server have a unique salt value for the same username-password pair. The salt also protects against off-line password guessing attacks.

The drawbacks of Lamport's Hash are that there is no mutual authentication and the user can only login a finite number of times before reconfiguration is required.

2.3.5 Authentication Using PKC

2.3.5.1 One-way Authentication Using PKC

Figure 2.12 shows how PKC can be used for one-way authentication. Alice starts by sending her username to Bob. Bob responds with a challenge. Note that so far this procedure is exactly the same as Figure 2.6. At this stage however, Alice encrypts the challenge with her private key and sends the response back to Bob. Bob decrypts the response with Alice's public key, thus confirming that Alice is in fact Alice, since she has the correct private key. This step differs from Figure 2.6 in that the encryption and decryption steps are carried out using PKC instead of SKC. This has the advantage that even if the key data base at Bob is compromised, it does not compromise the security of the system since all Bob stores are public keys.

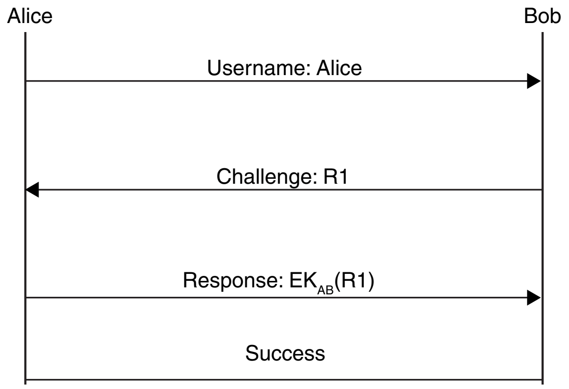


Figure 2.12: One-way Authentication Using PKC

Just as Figure 2.7 and Figure 2.8 were alternatives for Figure 2.6, similar alternatives may be used for Figure 2.12 provided the correct precautions are taken and the trade-offs understood.

2.3.5.2 Mutual Authentication Using PKC

Figure 2.13 shows how PKC can be used for mutual authentication. The procedure is a replica of Figure 2.10 except that instead of using one key (the shared secret) for encryption and decryption, the private key is used for encrypting the challenge to produce the response and the public key is used to decrypt the response and get back the challenge.

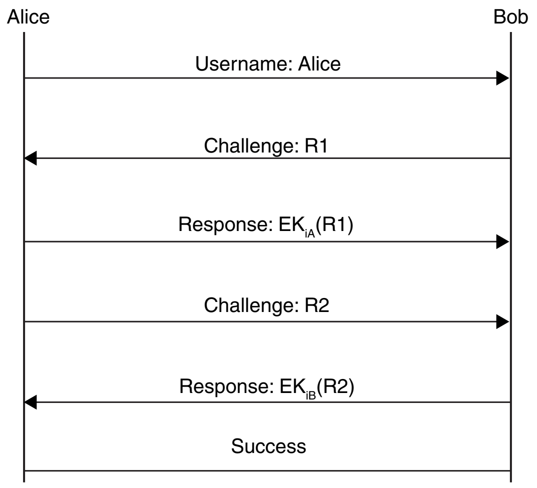


Figure 2.13: Mutual Authentication Using PKC

2.3.6 What to Use for Authentication: SKC or PKC?

SKC has the obvious advantage that it is much less computation intensive. This feature not only makes it much more “cheaper” and “lighter” but also makes it more resilient to DoS attacks.[19] However, there are two major drawbacks to using SKC for authentication. First, note that Bob needs to store the keys of all users that wish to talk to him in a database, or the system needs to employ a KDC as the central repository for all keys. In either case if the key database is compromised, the security of the whole system is compromised. Compare this to the PKC approach where the CA stores only the public key of all users and access to the CA database does not itself

[19] Less computation → more difficult to exhaust resources at Bob.

break the security of the system.[20] The second drawback of using SKC for authentication is that it allows Eve to collect <plaintext, ciphertext> pairs encrypted with the shared secret. Eve can do this by claiming to be Bob and sending the plaintext as a challenge to Alice. Alice would then respond with the encrypted challenge. The collected <plaintext, ciphertext> pairs database may then be used for dictionary attacks.

As we just saw, PKC does not suffer from two of the main drawbacks from which SKC suffers. However, PKC has its own disadvantages. First, PKC is a computationally intensive exercise making it a much less attractive option especially for portable devices which have limited computing power. Another problem which stems from this is that PKC usually makes the system much more stateful, making it more vulnerable to DoS attacks.

2.3.7 Session Hijacking

As we said at the beginning of this section, one of the primary uses of authentication in network security is to implement access control reliably. We looked at some authentication protocols which can be used for this purpose. However, all the protocols that we have discussed until now suffer from a class of attacks known as session hijacking. The underlying principle of session hijacking is that instead of trying to "break" the authentication protocol, it circumvents it completely. In other words, if Eve wants to hijack a session between Alice and Bob, she does not try to learn the password[21] of Alice. Instead, she lets Alice authenticate with Bob. When Alice has completed her authentication process successfully with Bob, Eve comes in and claims to be Alice.

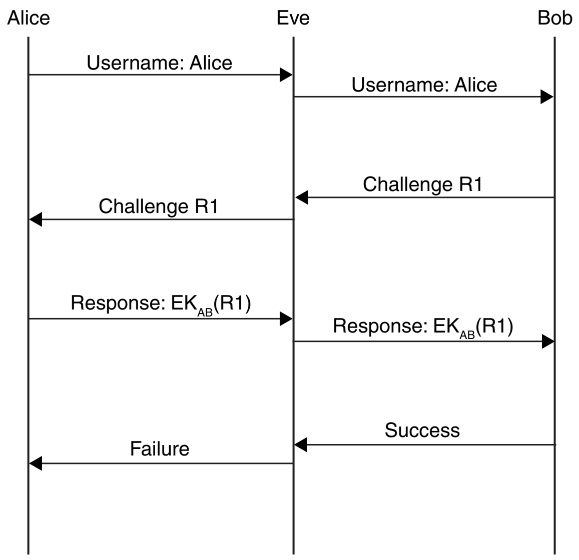


Figure 2.14: Session Hijacking

Figures 2.14 and 2.15 are two examples of session hijacking.

In Figure 2.14, Eve launches a very simple DoS attack. She blocks all "Authentication Success" messages

[20] System security is compromised if Eve can masquerade as the CA and issue fake certificates.
[21] Or the hash (password) or the key.

transmitted by Bob and instead replaces them with "Authentication Failure" messages. Since this message is not cryptographically protected, there is no way for Alice to know that the message came from Eve and not Bob. This is a simplistic form of session hijacking.

In Figure 2.15, Eve lets Alice and Bob complete the authentication. Once the final "Authentication Success" message is issued, Eve jumps in and claims to be Alice. Since the session messages are not cryptographically protected, there is no way for Alice to know that the messages came from Eve and not Bob.

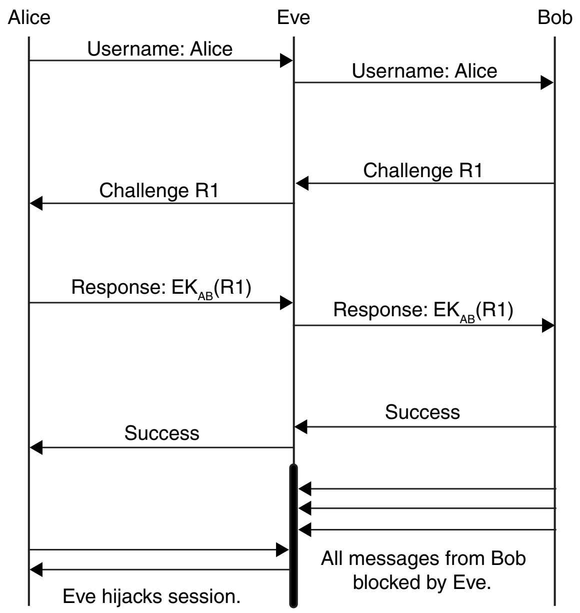


Figure 2.15: Session Hijacking—*Variation*

Session hijacking exploits the fact that the authentication protocols do not establish a security context; that is, there is no way to link the result of the authentication protocol to the rest of the session. What is needed to protect against such attacks is for the authentication protocol to be "linked" to the rest of the session; in other words the creation of a security context within which the rest of the session can be carried out. This can be achieved by ensuring that the execution of the authentication protocol automatically leads to the generation of a shared secret session key at both Alice and Bob. The session key can then be used to encrypt each message sent so that Alice can be assured that each message is in fact coming from Bob and vice versa. Such authentication protocols are sometimes referred to as context-establishing or key-generating authentication protocols. This is the approach taken in the two standardized authentication protocols we discuss in Sections 2.3.8 and 2.3.9.

2.3.8 Needham Schroeder

One of the classic SKC-based mutual authentication protocols using KDCs is the Needham Schroeder protocol, named after its designers. The protocol is shown in Figure 2.16a.

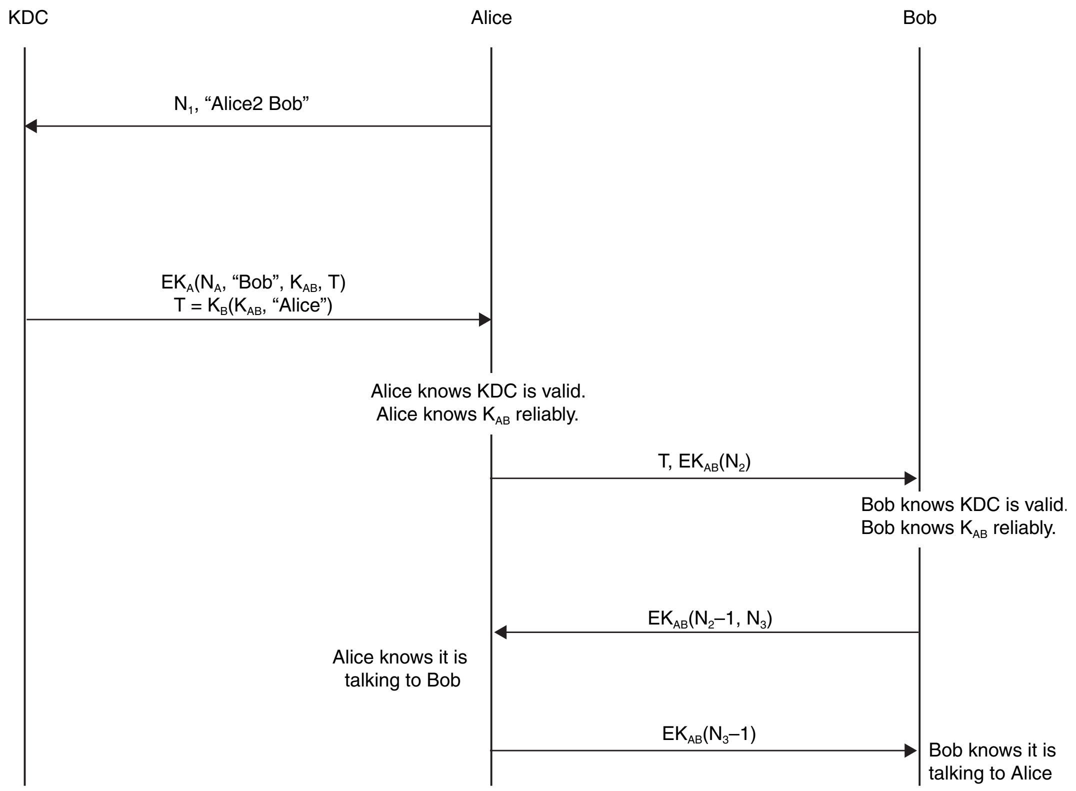

KDC
Alice
Bob
N_1, "Alice2 Bob"
$EK_A(N_A$, "Bob", K_{AB}, T)
T = $K_B(K_{AB}$, "Alice")
Alice knows KDC is valid.
Alice knows K_{AB} reliably.
T, $EK_{AB}(N_2)$
Bob knows KDC is valid.
Bob knows K_{AB} reliably.
$EK_{AB}(N_2–1, N_3)$
Alice knows it is talking to Bob
$EK_{AB}(N_3–1)$
Bob knows it is talking to Alice

Figure 2.16a: Needham Schroeder

It is important to keep in mind that the protocol is SKC-based; each user's key is known only to the user herself and the KDC. Also, we proceed with the assumption that this key has not been compromised.

The protocol starts with Alice sending a message to the KDC. This message contains a nonce (N1)[22] and information about (the "username" of) the originating and the terminating parties (Alice2Bob also known as A2B).

[22] A nonce is a number with the property that it will not be re-used, i.e., it's just used once; hence the name: nonce.

KDC responds to this message by sending EK_A{N1, “Bob”, K_{AB}, ticket} where ticket = EK_B(K_{AB}, “Alice”). Since the message is encrypted with Alice’s key (K_A), Eve cannot read the message. On receiving this message, Alice can be assured that she is talking to a valid KDC and not an impersonator since only the KDC (besides Alice herself) knows K_A. Therefore, Alice has now learnt the session key K_{AB} that she is supposed to use for authenticating with Bob. The next aim is to get this session key to Bob.

Alice then sends the ticket and EK_{AB}(N2) to Bob where N2 is another nonce (separate from N1) to Bob. On receiving this message, Bob knows that he can trust the ticket since it is encrypted with K_B which is known only to KDC and Bob himself. Note that even though this message has been sent by Alice, Bob trusts the ticket because it has been encrypted by the KDC. After decrypting the ticket, Bob knows K_{AB} and uses this to find out N2. Note that at this stage in the protocol, we have established a shared secret key K_{AB} between Alice and Bob. In the rest of the protocol, Alice and Bob use this shared secret to authenticate each other.

Bob then responds back with a message EK_{AB}(N2–1, N3), where N3 is yet another nonce. When Alice receives this message, it decrypts this message to determine N2 and compares it with its local copy of N2. If the values match, Alice concludes that she is talking to Bob, since only Bob (besides Alice) knows both K_{AB} and N2.

Finally, Alice sends EK_{AB} (N3–1). When Bob receives this message, he decrypts this message to determine N3 and compares it with his local copy of N3. If the values match, Bob concludes that he is talking to Alice since only Alice (besides Bob) knows both K_{AB} and N3.

2.3.9 Kerberos

One of the most used authentication protocols using SKC via KDC is the Kerberos protocol. The Kerberos literature defines the concept of principals. A principal may either be a user (client) or a machine (server). Each principal has its own master key which it shares with the KDC. For human users, the master key is derived from a password (for example, using a hash function). Figure 2.16b shows how the Kerberos protocol works.

The first half of the figure explains how a user logs in securely in to her workstation without the need for the workstation to store each user’s key. The keys of all users are stored at the KDC only. This improves system security because it does not assume a trust relationship between the node and the user. Therefore even if the node is compromise, the system security is not since the node does not store the key.

In Figure 2.16b, when Alice types in her username, the workstation relays the username to the KDC. The KDC replies back with EK_A\{SA, TGT\} where SA is a *session key* generated by the KDC and is unique for each login session and ticket granting ticket (TGT) is EK_{KDC}\{“Alice”, SA\}. The TGT therefore binds the session key with the username. When the workstation receives this message, it prompts the user to enter her password. When Alice types in her password, the workstation converts this password to K_A using a specified algorithm. The workstation then uses this K_A to decrypt the message it received from the KDC. If the password typed in was correct, the message from the KDC decrypts correctly and the workstation now knows SA and the TGT.

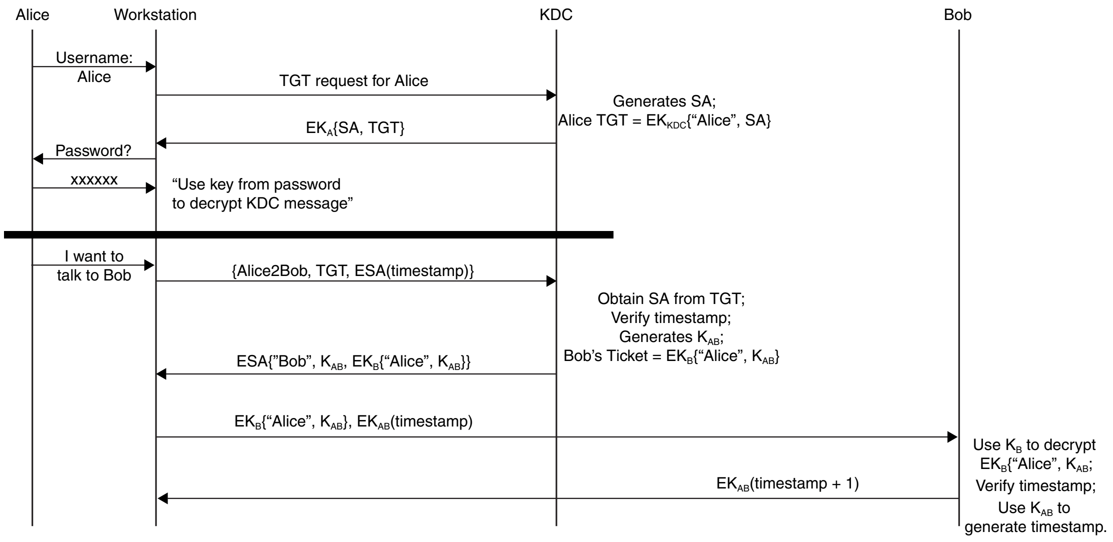

Alice
Workstation
KDC
Bob
Username: Alice
TGT request for Alice
Generates SA; Alice TGT = EK_{KDC}{“Alice”, SA}
EK_A{SA, TGT}
Password?
xxxxxx
“Use key from password to decrypt KDC message”
I want to talk to Bob
{Alice2Bob, TGT, ESA(timestamp)}
Obtain SA from TGT; Verify timestamp; Generates K_{AB}; Bob’s Ticket = EK_B{“Alice”, K_{AB}}
ESA{”Bob”, K_{AB}, EK_B{“Alice”, K_{AB}}}
EK_B{“Alice”, K_{AB}}, EK_{AB}(timestamp)
Use K_B to decrypt EK_B{“Alice”, K_{AB}; Verify timestamp; Use K_{AB} to generate timestamp.
EK_{AB}(timestamp + 1)

Figure 2.16b: Kerberos

Now, when Alice wishes to authenticate with another principal (say the server, Bob), Alice types Bob’s address into her workstation. The workstation then transmits A2B, TGT and SA (timestamp). When the KDC receives this message, it uses K_{KDC} to decrypt the TGT and obtain SA. The advantage of transmitting TGT in the message is that it relieves the KDC from having to store the session keys it generates for each user. Once the KDC has derived the SA from the TGT, it generates another session key, K_{AB}, to be used between Alice and Bob for this session. The KDC then uses SA to encrypt K_{AB} and sends it to Alice. The exact message it sends back is ESA\{“Bob”, K_{AB}, ticket\} where ticket = EK_B\{“Alice”, KAB\}. When Alice’s workstation receives this message, it uses the stored SA to decrypt it and obtain the K_{AB} and the ticket.

Alice then sends the ticket and EK_{AB}(timestamp) to Bob. Note that this is the first message that Alice has sent to Bob. When Bob receives this message he decrypts the ticket using his master key (K_B) to obtain K_{AB} and then uses K_{AB} to obtain the time-stamp transmitted by Alice. If the resulting timestamp matches the current timestamp at B (within reasonable limits), Bob is assured that it is in fact talking to Alice. This trust follows from the chain that only the KDC could have sent a valid ticket to Bob and since the ticket was valid, so must be K_{AB}, whose validity is certified by validating the timestamp.

Once Bob is assured that he is talking to Alice, he sends back $EK_{AB}(TS + 1)$ to Alice. On receiving this, Alice uses the stored K_{AB} to obtain the TS and compares it with the value that she had previously transmitted. If the timestamp value matches, Alice can be assured that she is in fact talking to Bob since only Bob (besides Alice and the KDC) could have known K_{AB}. At this point, Alice and Bob have completed mutual authentication.

2.4 Encryption Protocols

As we saw in Chapter 1, the primary use of encryption is to provide confidentiality. Encryption protocols take the message (plaintext) and a key as an input and generate an encrypted message (ciphertext). Obviously, the most desirable property of an encryption protocol is that it should be extremely difficult to obtain the plaintext from the ciphertext without the knowledge of the key. On the other hand, it should be "easy" to obtain the plaintext from the ciphertext if the key is known. There are other important desirable properties too. A good encryption protocol should hide patterns in the plaintext. Another desirable property is diffusion, which requires that a change

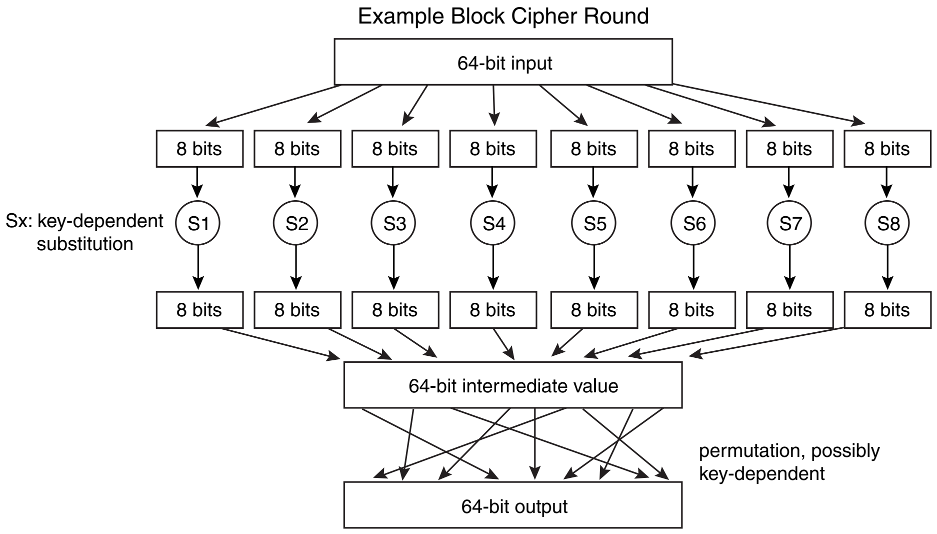

Example Block Cipher Round
64-bit input
8 bits
Sx: key-dependent substitution
S1
S2
S3
S4
S5
S6
S7
S8
64-bit intermediate value
permutation, possibly key-dependent
64-bit output

Figure 2.17: Example of a Block Cipher Round

in a single bit in the plaintext should change a large number of bits in the ciphertext. Yet another desirable property is that if a single bit is changed in the plaintext, it should lead to a change in a totally (apparently) unrelated set of bits in the ciphertext (confusion).

Encryption protocols (especially block encryption protocols) have traditionally been designed by mathematicians and the reasons behind their designs are probably known to a very small group of people. Most people treat encryption protocols like black boxes and with good reason. To see why, let us look at one of the most common ways to build a secret-key block cipher.

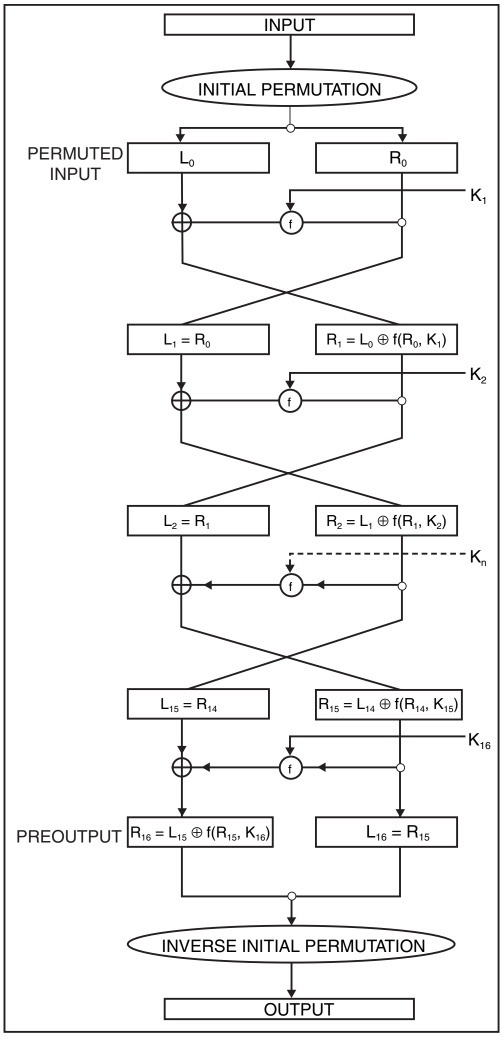

INPUT
INITIAL PERMUTATION
PERMUTED INPUT
L_0
R_0
K_1
f
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
K_2
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
K_n
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
PREOUTPUT
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$L_{16} = R_{15}$
INVERSE INITIAL PERMUTATION
OUTPUT

Figure 2.18: DES

This mechanism is shown in Figures 2.17 and 2.18. We start by breaking up the input block into manageable sized chunks (say 8 bits). On each of these chunks we do a substitution. A substitution operation refers to using the input chunk as an index into a predefined table and obtaining an output. The input value is then substituted with the value obtained from the table. This new value is then taken and made to run through a permuter (permutation). A permuter basically shuffles the bits of its input to generate an output. This completes one round of operation. Most encryption protocols consist of multiple rounds of such operation so that each bit winds up as an input to each of the substitutions. The multiple rounds ensure that the cipher has enough diffusion. Because of these multiple rounds, such block ciphers are also known as iterated block ciphers.

The process described above is ideal for what it aims to achieve: confusion and diffusion. It may seem simple at first but to realize its complexity, appreciate the fact that the process involves not only "garbling up" messages to make them meaningless to Eve but also of the reverse process of "de-garbling"—that is, obtaining back the plaintext message from the ciphertext at Bob.

The difficulty in analyzing most encryption protocols arises from the difficulty in understanding the design of the use of permuter and the substitution table (also known as an S-box). Consider for example, questions like the following. Why does the permuter move bit 3 to position bit 7 and not to position bit 1? Similarly, why is the S-table designed the way it is? These are all good questions but beyond the scope of this book. Also, most people don't worry about these details as long as they are assured that using the encryption algorithm will protect their information.

2.4.1 DES

We will look first at digital encryption standard (DES). For a long time, DES was one of the most widely deployed block encryption algorithms. It was a well designed protocol for the times in which it was designed. DES uses a 56-bit key for encryption. This was secure for a long time, but with the exponential growth in processing power per dollar (or pound) a 56-bit key is no longer considered secure. More about this in the next section. First, we will look at how DES works.

DES is designed around the concept of Feistel networks. Feistel networks are a type of iterated block cipher where each "round" can be described as $L_i = R_{i-1}$ and $R_i = L_{i-1}$ XOR $f(R_{i-1}, K_i)$. The function f is known as the mangler function and it can be as complicated as we want.

DES is a specific case of a Feistel network with a standardized mangler function.

DES uses a 56-bit key (actually a 64-bit key but out of the 64, 8 are parity bits) which it expands to generate sixteen 48-bit-per-round keys by taking a different 48-bit subset (different number of left shifts) of the 56 bits for each of the keys. DES has the advantage that decryption consists simply of running the algorithm with the reverse key order. The initial and final permutation (reverse of each other) used in DES do not use the key and do not add any additional security to the algorithm. They were probably added to make software implementation of DES slow and to discourage attacks. The shortcomings of DES included a short key and that it was complementary in other words If EK(P) = C, then EK(P') = C' where C' represents bitwise inverse. This means a brute force attack had to test just 2^{55} keys.

2.4.2 TripleDES or 3DES

Since the biggest drawback of DES is the limited key size, various attempts have been made to increase the "effective" key size by encrypting multiple times. The first option is to encrypt the plaintext twice with the same key; i.e., $C = EK_1(EK_1(P))$. Since we are still using only one key, the key space is still only 2^{56} keys large. This means that a brute force attack would have to work (only) twice as much for breaking the cipher. However, the communicating parties would have their work doubled too. Therefore, this is not a great solution.

The next option is encrypting the plaintext twice with two different keys, i.e., $C = EK_2(EK_1(P))$. This makes the key space 2^{112} keys large and therefore the cipher is as secure as using a 112-bit key. However, this increased security has been achieved only against brute force attacks. The double encryption DES is open to the meet-in-the-middle attacks which break it in 2^{n+1} attempts, where n is the key size (56 in case of DES).

P_g	K_0	C_0
P_g	K_1	C_1
P_g	K_2	C_2
.	.	.
.	.	.
P_g	K_n	C_n

Table 1

C_g	K_0	P_0
C_g	K_1	P_1
C_g	K_2	P_2
.	.	.
.	.	.
C_g	K_n	P_n

Table 2

Figure 2.19: Meet-in-the-Middle Attack Against Double-DES

Here is how this attack works. First, we assume that Eve has access to some <plain-text, ciphertext> pairs. Eve encrypts the plaintext from the given pair (P_g) with all possible 2^{56} keys to construct Table 1 with 2^{56} entries ($C_0 – C_N$) where $N = 2^{56}$. She then compares the corresponding ciphertext (C_g) with the values $C_0 – C_N$. A match leads to the discovery of a possible encryption key. Note that we say a "possible" encryption key since there may be multiple instances of C_g in the set $C_0 – C_{N.}$ If there is more than one match, we use another <plaintext, ciphertext> pair and repeat till we find the unique key which satisfies all <plaintext, ciphertext> pairs.

Eve then repeats the same process to find the decryption key. She decrypts the given ciphertext (C_g) from the given pair with all possible 2^{56} keys to construct Table 2 with 2^{56} entries ($P_0 – P_N$) where $N = 2^{56}$. She then searches this table to find matching entries in column 3. A match leads to the discovery of a possible decryption key.

This attack is not really practical since it requires huge tables requiring a total space of 100,000 terabytes. However, the theoretical existence of the attack was enough to warrant the next step: three steps of encryption and decryption using two keys.

The standardized way of doing multiple encryptions using DES is triple encryption. This is commercially known as 3DES and it works by using $C = E_{k1}(D_{k2}(E_{k1}(P)))$ and $P = D_{k1}(E_{k2}(D_{k1}(C)))$. Why are three rounds used instead of two? Why we do EDE instead of EEE or DDD? The answer to both these questions is: backward compatibility. Since E and D are equally easy (or equally hard) to do, the reason for choosing EDE is that if EDE is done with K1 = K2, then we have the "normal" (single) DES. Another reason for doing EDE as opposed to EEE is that if E is followed by E, then the initial permutation before the second E will reverse the final permutation after the first E. Note that 3DES is just a scheme for encrypting a single block. For encrypting a whole message, we would need to break it down into blocks and use modes as we do with DES.

Here again we have an option. We can either do the "chaining" of CBC on the inside or the outside. 3DES does chaining on the outside which means that we start with a block, triply encrypt it and then XOR it with the next plaintext block. The other option of doing cipher block chaining (CBC) on the inside involves encrypting the whole message with K1, doing CBC, taking the whole result, completely decrypting the message with K2, doing CBC and then taking the whole result and completely encrypting the message with K1.

With CBC done on the inside, an error (or an intentional modification by Eve) in ciphertext block n corrupts all future plaintext blocks. Is this desired? Yes and no. Yes because it makes the system more secure and no because sometimes people want the network to be self-synchronizing so that we can recover from transmission errors. An advantage of using CBC on the inside is that in this case it is possible to use three times as much hardware and pipeline the encryption so that it is as fast as a single encryption—this is not possible with CBC on the outside.

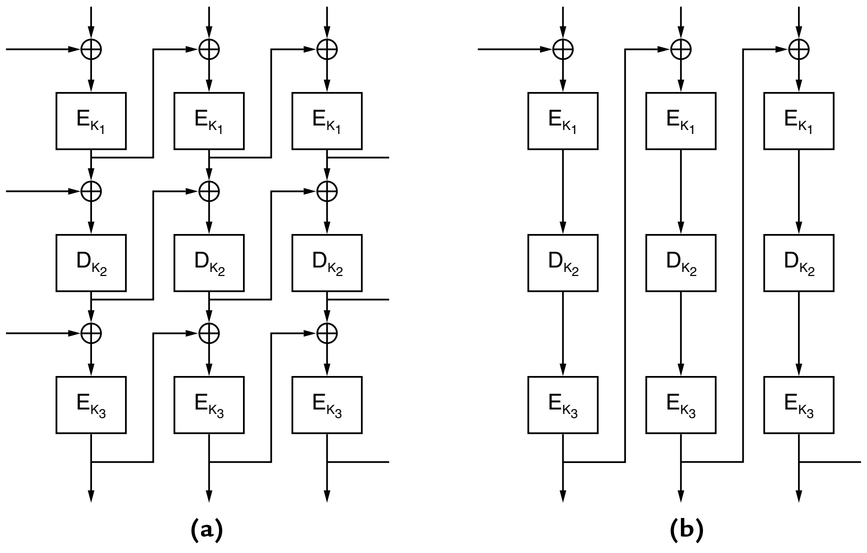


Figure 2.20: Triple Encryption in CBC Mode—(a) Inner CBC; (b) Outer CBC

2.4.3 AES

The advanced encryption standard (AES) algorithm converts a block of 128-bit plaintext into a block of 128-bit ciphertext. The algorithm uses a key which can be 128, 192 or 256 bits long. Increasing the key size used in the algorithm not only offers a larger number of bits to scramble the data, but also increases the complexity of the algorithm. This explanation will consider only the 128-bit key size for simplicity. With 128-bit keys, the AES algorithm core runs for ten rounds. A typical round is shown in Figure 2.21a and the complete algorithm is shown in Figure 2.21b. We start by considering a 128-bit data block for encryption. Think of the data block as an array of bytes where b_i refers to the *i*th byte with *i* ranging from 0 to 15.

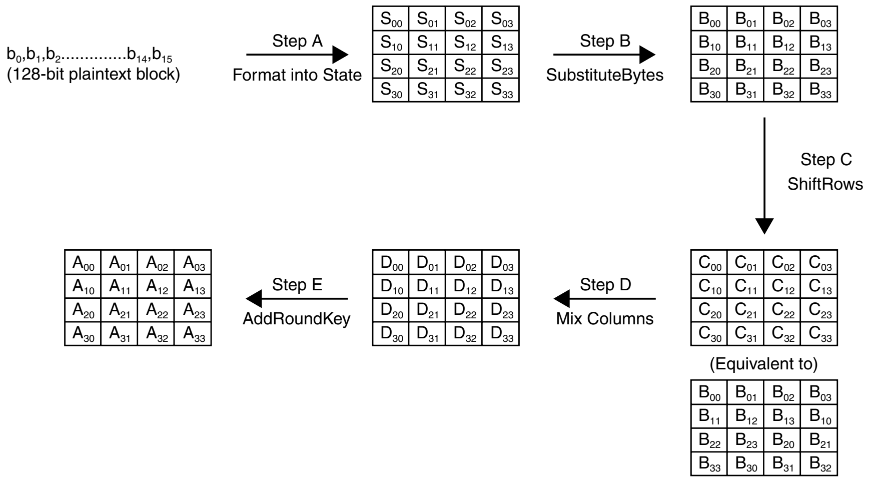


Figure 2.21a: AES Steps

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Figure 2.21b: AES Algorithm

Step A converts the byte array into a state matrix with b_0– b_3 forming column 1, b_4–b_7 forming column 2 and so on. Meanwhile the 128-bit key undergoes a key-expansion algorithm to generate 11 (number of rounds + 1) 128-bit keys numbers K_0 to K_{10}. Each of the new keys can be thought of as a set of four 32-bit values.

Step B substitutes each byte in the matrix with a new value. To find the new value for A_{xy}, it breaks A_{xy} into two nibbles and does the following:

$$B_{xy} = S\_Box[A_{xy\_high}][A_{xy\_low}]$$

where S_Box refers to a 2-D 16*16 byte lookup table defined in the AES standard.

Step C switches the order of the column entries for each row in the matrix. The nth row gets shifted to the left by n elements. So row 0 remains as is, row 1 shifts left by one, row 2 by two and row 3 by three elements.

Step D operates on each column, one column at a time. It mixes the value of any given column with other values from the column to obtain a new column. It does so by multiplying (in a Galois Field) the given column by a 4*4 byte matrix as shown in Figure 2.22.

Step E performs an XOR of each column with a 32-bit value from the key for this round (K_0 in this case). If c is the column number, Step E does the following:

$$\{A_{0c}, A_{1c}, A_{2c}, A_{3c}\} = \{C_{0c}, C_{1c}, C_{2c}, C_{3c}\}\ XOR\ w_r[c].$$

where w_r is the key for that round represented as a 4-element array of 32 bits.

$$\begin{bmatrix} D_{0c} \\ D_{1c} \\ D_{2c} \\ D_{3c} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} B_{0c} \\ B_{0c} \\ B_{0c} \\ B_{0c} \end{bmatrix}$$

Figure 2.22: Step E Matrix in AES

The output of Step E becomes the input to the next round of the AES cipher. For a 128-bit key, this process continues for ten rounds and the final output is the ciphertext.

2.4.4 RC4

A stream cipher encrypts data 1 byte at a time. RC4 is a variable-key-size stream cipher, which means that RC4 does not place any limit on the size of the key which it takes as an input. RC4 works in OFB.[23] RC4 is used to generate the pseudorandom

[23] See Section 1.4.4 to see how OFB works.

string used in OFB. The RC4 algorithm takes the shared key as an input and generates a pseudorandom string independent of the plaintext.

We start by creating an S-Box whose elements are a function of the key. To encrypt a given stream of plaintext, we generate a 1-byte "key-byte" for each byte in the plaintext as shown in Figure 2.23.

```
Initially, $i = j = 0$
$i = (i + l) \bmod 256$
$j = (j + S_i) \bmod 256$
swap $S_i$ and $S_j$
$t = (S_i + S_j) \bmod 256$
$S_{key\text{-}byte} = S_t$
```

Figure 2.23: RC4

On the sender side, the byte $S_{\text{key-byte}}$ is XORed with the plaintext byte to obtain the ciphertext byte. On the received byte, the same $S_{\text{key-byte}}$ is generated (since the key is shared and since the sender and the receiver are synchronized) and XORed with the ciphertext to obtain the plaintext.

2.5 Integrity Protocols

2.5.1 CBC Residue

CBC residue is an interesting way of using the CBC mode of encryption to achieve message integrity. Refer to Section 1.4.1 to see how CBC is used for achieving message confidentiality. Compare that with Figure 2.24.

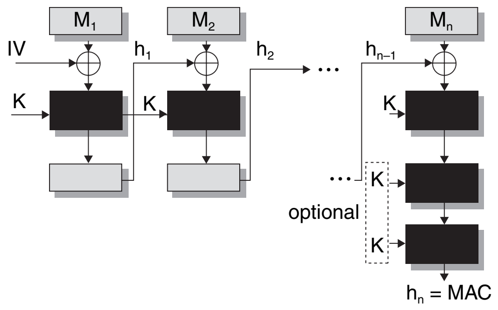


Figure 2.24: CBC Residue for Message Integrity

For calculating the CBC residue, the first block of the message is XORed with an IV and then encrypted with the secret key. The resulting cipher text is XORed with the next block and the result is again encrypted with the secret key. This process is chained along the message blocks. The end result is the CBC residue which is as long as one message block. The important thing to note here is that the calculated CBC residue depends on each and every block of the message. If any one of these blocks were to be modified by Eve, the CBC residue would be out of sync with the message. In effect, the CBC residue is a cryptographic snapshot of the whole message and can therefore be used by the receiver to verify the integrity of the message.

2.5.2 CRC32

Section 1.2.2 explains how hash functions are used to achieve message integrity. Hash functions can be divided into two categories: linear and nonlinear. A linear hash function is one which satisfies the following property:

$$f(x \oplus y) = f(x) \oplus f(y), \text{ where } \oplus \text{ represents a XOR operation.}$$

In this section we look at CRC-32, which is a linear hash function. The next section looks at the working of a nonlinear hash function. CRC-32 extends the concept of parity checking to achieve message integrity. The most basic form of parity checking works by extending the length of the message by one bit. The value of this bit is set so as to ensure that the total number of 1's in this message is even[24] (even parity).

Suppose we want to calculate the hash of a message M of length k bits. We follow the procedure shown in Figure 2.25:

$$\begin{gathered} M = i(x). \\ j(x) = i(x)\,.x^{32} \\ j(x) = g(x).q(x) + r(x) \\ t(x) = j(x) + r(x) \end{gathered}$$

Figure 2.25: CRC32

Step 1: Represent the message as a polynomial of order k where the coefficient of x^n can be 0 or 1 corresponding to the value of the nth bit in the message. We call this the information polynomial $i(x)$. So, we have the M represented as:

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots\ldots\ i_1x^1 + i_0$$

Step 2: Multiply $i(x)$ by x^{32} to obtain $j(x)$. This is equivalent to left shifting $i(x)$ by 32 bits and filling in the lower 32 bits with 0s. We now have:

$$j(x) = i(x)x^{32}$$

[24] Or odd for odd parity.

Step 3: Divide $j(x)$ by $g(x)$ to obtain a quotient $q(x)$ and a remainder $r(x)$ where $g(x)$ is the publicly known generator polynomial:

$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1.$$

Step 4: Add $r(x)$ to $j(x)$ to obtain $t(x)$. To use CRC-32 for message integrity purposes, we transmit $t(x)$ instead of $i(x)$.

Note that the transmitted data $t(x)$ has a length of k + 32 where k is the length of the original data message. The higher k bits of $t(x)$ contain the data information bits and the lower 32 bits consist of the "checksum." The aim of the four steps of CRC-32 is to make the transmitted message exactly divisible by the publicly known polynomial $g(x)$. The receiver of the message verifies that the message it receives is in fact divisible by $g(x)$. If so, the receiver concludes that the message has not been modified in transit.

The problem with linear hash functions like CRC-32 is that a single corrupted bit in the message will result in a single bit change in the calculated CRC, but multiple corrupted bits may cancel each other out. It is due to this feature that it is comparatively easier to break the message integrity provided by CRC-32.

2.5.3 MD5

Recall from Section 1.2.2 that hash functions achieve message integrity by calculating a message integrity check (MIC) also known as message authentication code (MAC). The MIC is usually calculated over the sent message and "attached" to the message by the sender. On receiving the message, the receiver can calculate the MIC too and compare it with the received value of the MIC.

The calculation of the MIC is what is specified by the message integrity or hash protocol. MD5 is such a hash function. It is defined in RFC 1321. The executive summary of this RFC says: "The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 2^{64} operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2^{128} operations."

Let's suppose we want to calculate the hash of a block of data, D, which is b bits long. Figure 2.26 explains how MD5 works.

Step 1: Pad the data block to make it k bits long where k is the smallest number larger than b such that k = 512n – 64; n being any natural number.

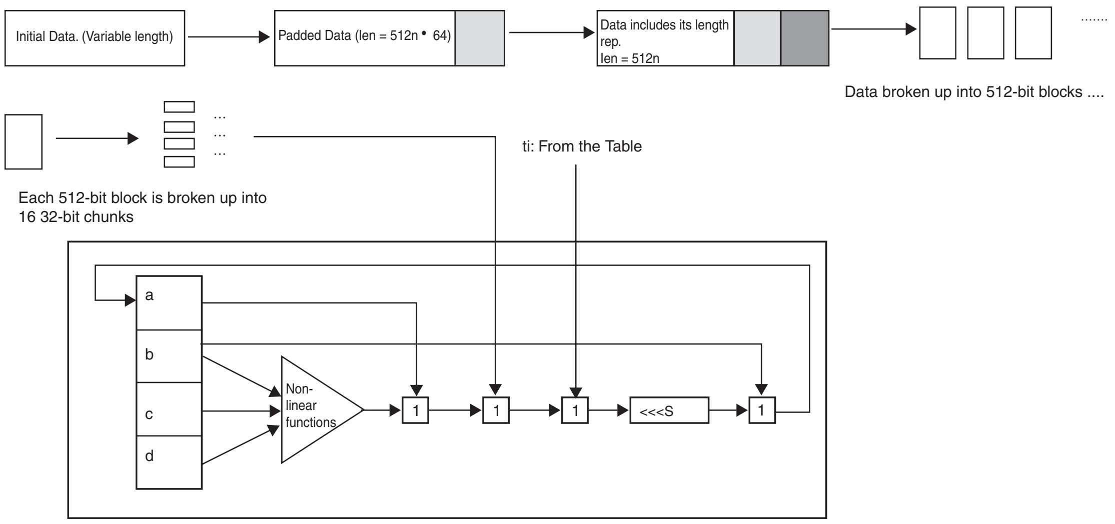

Initial Data. (Variable length)
Padded Data (len = 512n • 64)
Data includes its length rep. len = 512n
....
Data broken up into 512-bit blocks ....
Each 512-bit block is broken up into 16 32-bit chunks
ti: From the Table
a
b
c
d
Non-linear functions
1
1
1
<<<S
1

Figure 2.26: MD5

Step 2: Append to this, a 64-bit number which represents the length of the block. The resulting block has a length which is an exact multiple of 512.

Step 3: Now, break up this data block into 512-bit sub-blocks. Each 512-bit sub-block is then broken into sixteen 32-bit words to be fed to the MD5 operation. We refer to these words as a, b, c and d.

Step 4: Besides the data itself, MD5 uses four nonlinear functions F, G, H and I and a precalculated table of look-up values to calculate the hash of the message. The nonlinear functions are designed such that if the corresponding bits of b, c and d are independent and unbiased, then each bit of the result will also be independent and unbiased. Herein lies the strength (nonlinearity) of MD5.

The first MD5 operation starts with a, b, c and d initialized to values predefined in the standard. The nonlinear function used is F. The first 32-bit word of the first 512-bit block is fed into the operation along with the appropriate look-up value from the table. The operation performs the nonlinear function over three of a, b, c and d. Then it adds the result to the fourth variable, the 32-bit supplied word and the value from the look-up table. Then it rotates this value a variable number of bits and adds the result to one of a, b, c or d. For each 512-bit block, sixteen such operations are carried out to cover the entire block. At this time, the[25] 512-bit block is said to have finished a round.

[25] Unless of course, the encryption protocol was compromised.

The values from the first round are fed to the second round, which carries out sixteen similar operations using the nonlinear function G. Similarly, the third round uses H and the fourth round uses I. The output of the fourth round is finally added to update the initial values of a, b, c and d. The complete set of four rounds which operates on each 512-bit block is called a loop. Obviously, there are as many loops in the MD5 hash as there are 512-bit blocks in the original message.

In Section 1.2.2, we also said that the calculation of the MIC to be used for message integrity does not require the use of a shared secret. This is true, but leaves open the possibility of an attack. Consider what happens if Eve captures a message containing a MIC, alters the contents of the message as she wants and then updates the MIC appropriately. This is not difficult because the hash protocol is not secret. Arguably, if the message and the MIC were encrypted (for confidentiality), this attack would not be possible. But what if the users do not want to use encryption (for confidentiality). Rather all the users want to do is ensure the integrity of the message. This is the incentive for using keyed-hash functions.

Keyed MD5 works by computing the hash of (key | message). The reason we add the key before the message is that it makes the scheme more collision resistant than it would be if the key was appended to the end of the message. There is a subtle bug with this scheme which makes it susceptible to append attacks. Most hash algorithms (MD4, MD5, SHA-1) work by padding the message to a multiple of 512 bits and then digesting it left-to-right in 512-bit blocks. This means that all we need to compute the message digest through chunk n is the message digest through chunk n-1 plus the chunk n. Carol can launch an append attack by exploiting this property. Suppose Carol doesn't care what is in the message. All she is interested is in appending a block to the end of the message. Since she already has M and H(M), she adds her block to the end of M to get M' and calculate H(M') by doing H(H(M), new-block).

We can protect against this attack either by calculating MIC as hash (key | message | key) or simply as hash (message | key). Since the key is at the end of the MIC, Carol cannot add the message block after that. Another way to protect against the append attack is to define MIC as a subset of bits obtained by taking the hash (key | message). This works because Carol can no longer compute the new MIC as before.

CHAPTER 3

Security and the Layered Architecture

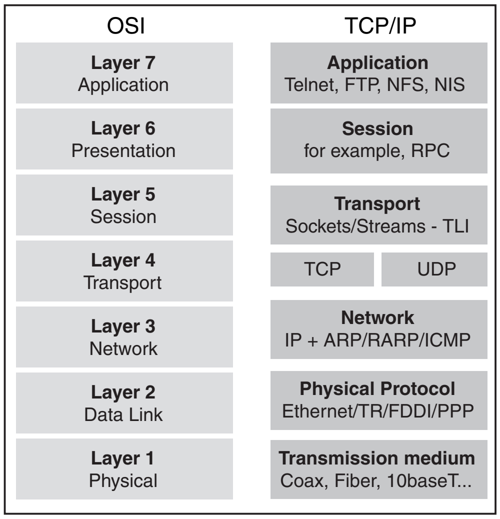
3.1 Introduction

We have now looked at the core concepts and some of the most popular protocols used in network security. There is however, an architecture question which still needs to be answered. Most network protocols are based on the concept of a layered architecture where each layer is dedicated to fulfill a certain function. The question now is: which layer is responsible for providing network security?

The answer depends on which network we are talking about. In the next few chapters we will look at three different kinds of networks: traditional wireless networks (voice-oriented), wireless local area networks (data-oriented), and wireless ad hoc networks. Since most networks in the near future are envisioned to be IP-based, we look at the layered architecture of IP networks in this section.

Figure 3.1 shows the layered architecture of IP-based networks. The IP layer is the glue layer which holds together the various link layer and transport layer protocols. Each layer implements a specific set of functionality which is used by the higher layer(s).

Figure 3.1: The OSI Layer Model


OSI
TCP/IP
Layer 7 Application
Layer 6 Presentation
Layer 5 Session
Layer 4 Transport
Layer 3 Network
Layer 2 Data Link
Layer 1 Physical
Application Telnet, FTP, NFS, NIS
Session for example, RPC
Transport Sockets/Streams - TLI
TCP
UDP
Network IP + ARP/RARP/ICMP
Physical Protocol Ethernet/TR/FDDI/PPP
Transmission medium Coax, Fiber, 10baseT...

From a security prospective, the question is where to implement the various security algorithms. Unfortunately, there is no single correct answer to that. There is however an ideal answer—at every layer. Implementing security at every layer may appear to be redundant but a closer look would reveal the need for it.

Let's start from the lowest layer. If the physical layer provides security, does any other layer need to implement security?

3.2 Security at Layer 1

Layer 1 deals with the physical transmission of the bits over the medium and covers topics like channel coding and modulation. Traditionally, security has not been a domain of Layer 1 protocol planners. However, in the wireless world, the advent of spread spectrum protocols does provide a certain amount of security at the physical layer. Spread spectrum protocols like Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS) were designed with the aim of minimizing the effects of interference from narrow band sources. DSSS works by using orthogonal spreading codes to spread the signal. What this means is that to transmit 1 bit of data the protocol in fact transmits 8 bits (referred to as chips). This has the effect of increasing the signal bandwidth eight times. It is this increase in the signal bandwidth transmission which protects it from narrow band interference. The chip sequence used to convert a single bit of data to 8 bits of transmission is what provides the inherent security in the protocol. Arguably, if an eavesdropper does not know the chip sequence, he cannot access the wireless signal. Similarly, FHSS spreads the signal by continuously changing the carrier frequency (frequency hopping). The frequency hopping sequence (that determines which carrier frequencies to use next) is effectively "the code" here. If an eavesdropper does not know the frequency hopping sequence, he cannot physically access the wireless signal.

It is important to emphasize here that these protocols provide no cryptographic protection. The "security" provided by these protocols stems from keeping the codes (chip sequence/frequency hopping sequence) secret. Since the codes themselves are not cryptographically protected and are usually well known (or easy to figure out), the security is often enough only to keep out the most casual of eavesdroppers. It is therefore better to think of physical layer security as a deterrent, especially against denial of service (DoS) attacks (think frequency-jamming).

3.3 Security at Layer 2

Layer 2 is concerned with establishing link-layer connectivity; that is, connecting devices to each other. Again, Layer 2 protocols traditionally have not had any security built into them. The design objective of IP (the most dominant Layer 3 protocol today) was to bind together various Layer 2 protocols so that higher layer protocols could be designed independently of the underlying channels. It is therefore instinctive to implement security at the IP layer rather than trying to design protocols for each link layer independently. In the wired world, the biggest deterrence at Layer 2 is physical access to the media.

Consider a local area network (LAN) designed using 802.3 (Ethernet). This LAN would consist of the physical wires, switches and hubs. To connect to such a LAN, a user would typically have to physically "plug-in" his computer into a switch or a hub. In other words, a user needs physical access to the network to connect to it. This requirement of having physical access to the network to connect to it serves as a security mechanism at Layer 2. This is not to say that there are no security protocols used in wired networks at Layer 2. We will see one such protocol in a moment. The point is that wired networks have an inherent security mechanism (physical access) which is missing in wireless networks.

3.3.1 Extensible Authentication Protocol (EAP)

Even though wired networks offer an inherent security hurdle by requiring physical access to the network, this alone is in no way a sufficient protection mechanism. Several Layer 2 security protocols have been designed, but probably the most relevant one for our purposes is EAP. To understand the design and the motivation behind EAP, a little background is necessary.

The most popular way of connecting to the internet today is to dial in over the phone line using a modem. The protocol used for this is Point to Point Protocol (PPP). From a system architecture view point, the service provider maintains a Point of Presence (PoP) in every geographical region where it wishes to provide service. The PoP acts like a network switch connecting user modems to the Internet. However, before allowing any user connection to the Internet, the service provider would like to authenticate the user. Since any user should be allowed to connect to the Internet from any geographical area and since each PoP cannot be expected to maintain the credentials of all subscribers, we have to offload the authentication process to a remote site which maintains a central repository of all user credentials. This is the authentication server. Now, we have three parties involved in the authentication process: the supplicant (the user trying to gain access), the authenticator (the PoP, which acts as the direct point of contact to the supplicant and controls access to the network) and the authentication server (which is responsible for actually authenticating the supplicant). The authentication server is therefore the decision-maker and the authenticator is the one which implements this decision. The communication protocol used between the authenticator and the authentication server is Remote Access Dial-In User Security (RADIUS). This system architecture is shown in Figure 3.2.

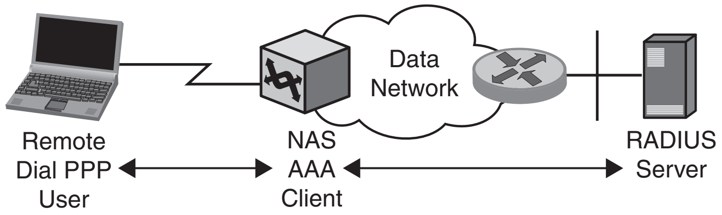


Figure 3.2: Authentication Model for Dial-In Internet Access

The PPP protocol specifies two authentication protocols that can be used: Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP). The former uses a username-password-based scheme where the usernames and the passwords are transmitted in plaintext. Obviously, it is not a very secure protocol. The latter uses a challenge-response-based mechanism which is much better than PAP, but still not considered strong enough, especially given the processing power available today. The problem is that anytime someone wishes to use a different authentication method with PPP, they have to go register the authentication method with Internet Assigned Numbers Authority (IANA). This is not as easy as it sounds, since the IANA has to keep in mind the existing the PPP compliance systems. It is to get around this problem that the EAP was designed.

The EAP (RFC 2284, RFC 2284bis) is not really an authentication protocol. It is actually a transport framework which runs over link layer protocols and supports multiple authentication mechanisms. The availability of this framework means that once EAP has been implemented over the link layer in a given setup, various authentication protocols (MD5, TLS and so on) can be plugged into it dynamically without affecting the rest of the system architecture.

The EAP framework is peer-to-peer and is based on requests and responses. An EAP-based authentication usually starts with the authenticator sending an identity request to the supplicant. On receiving this request, the supplicant replies with an identity response which may contain its login, username, MAC and so on.

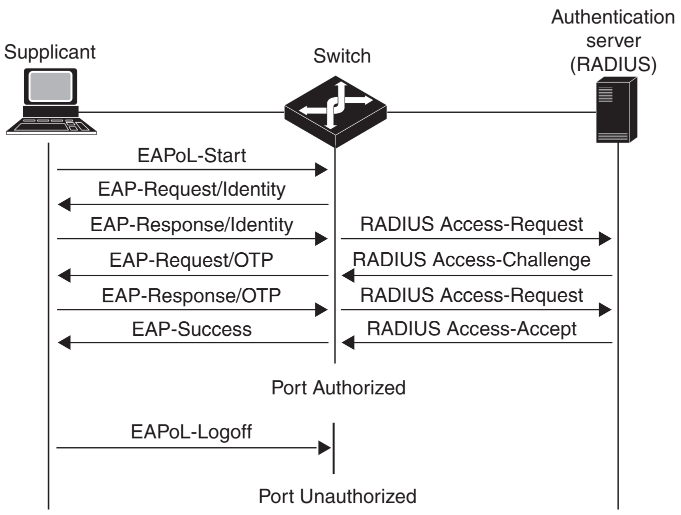


Figure 3.3: The EAP Architecture

On receiving this message, the authenticator (switch) reformats this message as a RADIUS-Access-Request message and forwards it to the backend authentication server. The authentication server now starts the authentication process using an appropriate authentication protocol. The messages sent by the authentication server are simply reformatted as EAP messages and forwarded to the supplicant by the authenticator. Note the role of the authenticator as a "pass-through" during the authentication process.

Once the authentication process completes, the RADIUS server issues a RADIUS-Access-Accept message. On receiving this message, the authenticator not only forwards it to the supplicant but also opens the "controlled port" to allow the supplicant access to the network. The network port is now said to be in authorized state. On the other hand, if the RADIUS server issues a RADIUS-Access-Reject message because of an authentication failure, the authenticator does not open the controlled port, thus restricting the supplicant from accessing the network.

The use of EAP has several advantages. First, EAP allows any arbitrary authentication protocol to be exchanged between the supplicant and the authentication server. Second, the authenticator does not have to understand each authentication method and may act as a passthrough agent for a back-end authentication server. This separation of the authenticator from the backend authentication server allows for higher flexibility and simple, low-cost authenticators. Additionally, this separation simplifies key and credentials management by concentrating this function at a back-end server. All these features make EAP ideally suited for PPP (and also for 802.11, as we shall see later).

However, there are several drawbacks of EAP too. First, even though EAP is peer-to-peer and supports authentication in both directions (A may authenticate B and B may authenticate A), there is no inherent mechanism in EAP to tie the two authentications together as part of a session.

Second, EAP does not provide protection against a forged "EAP-success." Since the success message is not cryptographically protected, it may be forged by a malicious Eve, thus nullifying any previous authentication procedure.

Third, EAP does not provide any mechanism to link the authentication procedure to the following session. It leaves this as the responsibility of the link layer stating that "…. it is necessary for the lower layer to provide per-packet integrity, authentication and replay protection that is bound to the original EAP authentication …"

3.3.2 EAPoL: EAP Over LAN

Though originally designed for PPP, EAP can also used in traditional LANs with the end-user's computer serving as the supplicant, the switch or hub serving as the authenticator and a backend server being the authentication server. IEEE 802.1X (EAPoL) specifies how to do this.

The standard specification of IEEE 802.1X (henceforth referred to as 802.1X) states that "(it) …is a mechanism for port-based network access control that makes use of the physical access characteristics of IEEE 802 LAN infrastructure in order to provide a means of authenticating and authorizing devices attached to a LAN port that has point-to-point connection characteristics, and of preventing access to that port in cases which the authentication and authorization process fails."

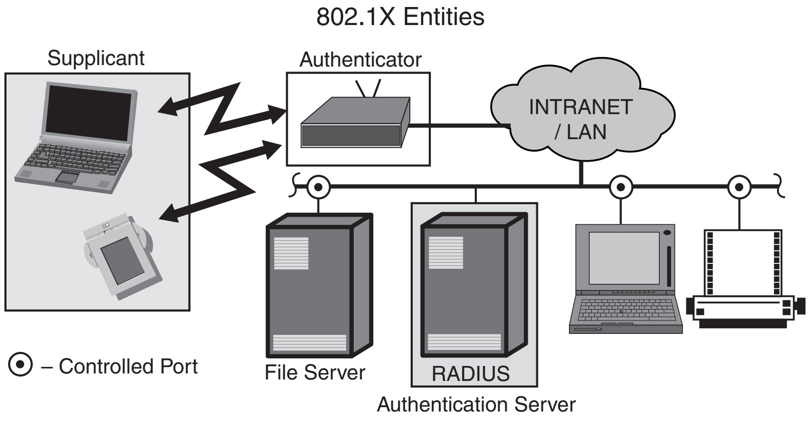


Figure 3.4: EAP Over LAN

In simpler terms, 802.1X specifies how to implement EAP over IEEE 802 LANs. For this reason, 802.1X is also referred to as EAP over LAN (EAPoL). In 802.1X, the supplicant requests access from an authenticator. The authenticator creates two ports for this supplicant—the controlled port and the uncontrolled port (see Figure 3.5).

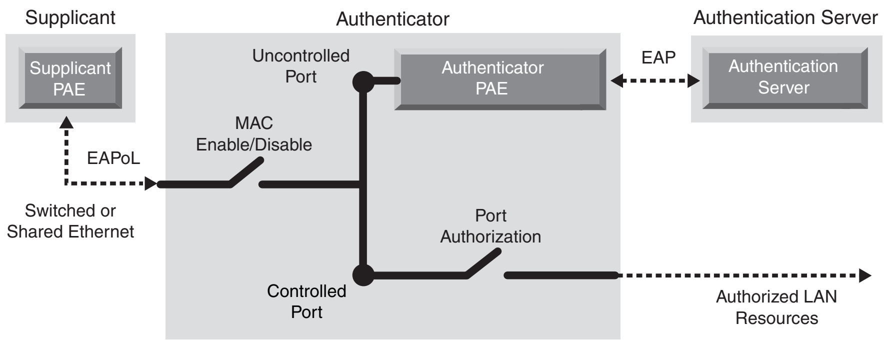


Figure 3.5: Ports in EAP model

It is important to note that the "ports" being mentioned here are logical entities. The controlled port is initially kept in a closed state and is opened only when the authentication server instructs the authenticator to do so. The authentication process between the supplicant and the authentication server is carried out over the uncontrolled port. The uncontrolled port works by allowing only EAP packets from the supplicant to pass through into the network. The EAP packets received by the authenticator are passed through to the authentication server. If the authentication server is a physically separate device, the authenticator relays these packets using RADIUS. When the authentication server completes the authentication process, it sends an "accept" or "reject" message to the authenticator and the supplicant depending on the outcome of the authentication process. If the authentication process was successful, the authenticator opens the controlled port, thus allowing the supplicant access to the network.

Note that in Figure 3.5, the authenticator combines the use of EAPoL with address-based authentication mechanism as mentioned in Section 2.3.1.

3.3.3 EAP-TLS: TLS Handshake Over EAP

Authentication methods may be divided into two broad categories: those which result in the establishment of a security context[1] at the end of the authentication process (TLS and so on) and those which do not (MD5, SHA and so on). Since EAP in itself

[1] For an explanation of the term *security context*, see Section 2.3.7

does not provide for an establishment of the security context, it is usually prudent to use EAP with an authentication protocol which establishes a security context. EAP-TLS is such an example and we look at it in more detail here.

As explained in Section 3.5, Transport Layer Security (TLS) was initially (RFC 2246) designed to be a library of wrapper functions around the socket layer. However, RFC 2716 modifies TLS to sit over EAP and calls it EAP-TLS. Using TLS as the authentication protocol has the additional advantage that Alice and Bob do not need to have a preshared secret to authenticate each other: instead, such a premaster key is established at run time since TLS is a context-establishing/key-generating authentication protocol.

Figure 3.6 shows an EAP-TLS exchange using DH where mutual authentication is successful from the perspective of the supplicant (Alice) and authenticator (Bob). The backed RADIUS operations to the authentication server are not shown for simplicity. For details of the TLS messages contained inside the EAP requests/responses refer to Section 3.5.

EAP-TLS ensures that the master secret derived from TLS is not used during the session since this secret (read key) has already been used in a previous context (that

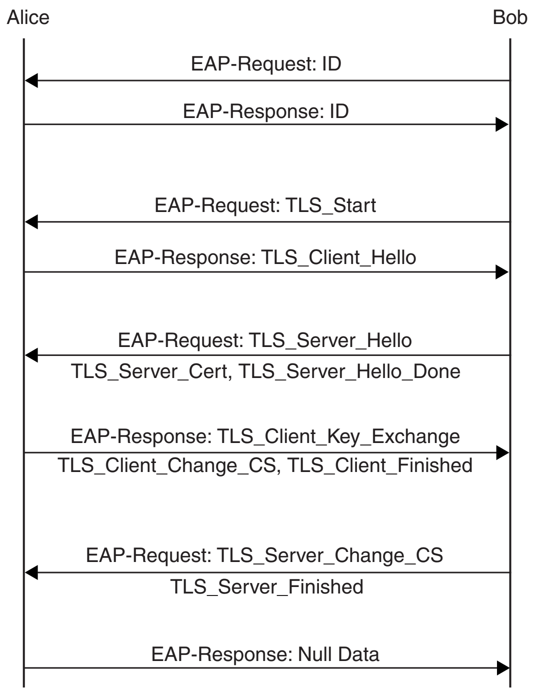

Alice
Bob
EAP-Request: ID
EAP-Response: ID
EAP-Request: TLS_Start
EAP-Response: TLS_Client_Hello
EAP-Request: TLS_Server_Hello
TLS_Server_Cert, TLS_Server_Hello_Done
EAP-Response: TLS_Client_Key_Exchange
TLS_Client_Change_CS, TLS_Client_Finished
EAP-Request: TLS_Server_Change_CS
TLS_Server_Finished
EAP-Response: Null Data

Figure 3.6: EAP-TLS

of the handshake session). Instead, new keys are derived from the TLS master secret as follows:

```
K1 (128 bytes) = PRF (master_secret, "client EAP encryption," ClientHello.random+ServerHello.random)
K2 (64 bytes) = PRF ("", "client EAP encryption", ClientHello.random+ServerHello.random)

Alice_to_Bob_Encryption_Key     = K1 (Bytes 0-31).
Bob_to_Alice_Encryption_Key     = K1 (Bytes 32-63)   or same as Alice_to_Bob_Encryption_Key.
Alice_to_Bob_Authentication_Key = K1 (Bytes  64-95)
Bob_to_Alice_Encryption_Key     = K1 (Bytes 96-127) or same as Alice_to_Bob_Authentication_Key.
Alice_to_Bob_IV                 = K2 (Bytes 0-31).
Bob_to_Alice_IV                 = K2 (Bytes 32-63).
```

3.4 Security at Layer 3

Layer 3 is responsible for providing end-to-end connectivity. Compare this with Layer 2 which provides link-layer connectivity. Perhaps an example would make things clearer. Consider what happens when you want to connect from your office computer to a website. Layer 2 is responsible for connecting[2] your computer to your office local area network (LAN) whereas Layer 3 is responsible for connecting[3] your computer to the web server.

The Internet Protocol Security (IPSec) protocol works at Layer 3 of the Open Systems Interconnection (OSI) protocol stack. Since Layer 3 of the network stack sits inside the operating system itself, IPSec code is bundled in the operating system too, and is therefore transparent to all applications. The advantage of this architecture is that the responsibility of security is removed from the application developer and can be centralized at the operating system level. Proponents of this architecture argue that centralizing security is better than modifying each application.

The IPSec protocol can be understood in two parts. First, we have the Internet Key Exchange (IKE) protocol, which is responsible for authentication and session key establishment between the two communicating parties. Second, we have Authentication Header (AH) and Encapsulating Security Payload (ESP): the IP header extensions which are used for carrying information that is needed to decrypt and verify the encrypted or integrity-protected IP packet.

Before we delve into the details of IKE, AH and ESP, let us take a brief overview of how IPSec works. Remember that the aim of IPSec is to protect the communication between two IP addresses. However, one IP address might be communicating with

[2] This includes transmission and reception of packets to and from the LAN.
[3] This includes routing of packets to and from your computer to the web server.

more than one IP address at any given time,[4] and each one of these sessions might be using IPSec. Furthermore, each of these IPSec sessions may be using separate security parameters. How then do we distinguish between the different IPSec sessions given that all of them are using the same IP address? To achieve this, each system implementing IPSec maintains a security association database.

The term Security Association (SA) refers to a cryptographically protected connection. A SA is unidirectional in that it specifies completely all the cryptographic information required in one direction of the communication. This includes information like the identity of the remote end, the cryptographic key being used, the cryptographic services and algorithms in use, the sequence number currently in use, and so on. Each entry in the SA database contains the SAs that are currently being used by this system. These entries are indexed by the destination address and the Security Parameter Index (SPI). The system searches the database using the destination address when it needs to transmit a packet to that destination address using IPSec. On the other hand, when the system receives an IP packet, it examines the AH/ESP header to determine the SPI and then uses this SPI to index into the SA database to find out all the information it needs to process this packet.

Authentication header (AH) and encapsulating security payload (ESP) are the two types of IPSec headers specified by IPSec. They are described in RFCs 2402 and 2406 respectively. Why does IPSec specify two header extensions? In theory, the two extensions serve two different requirements. In practice, ESP suffices for most practical purposes. AH provides support only for integrity protection of the payload in the packet and for some fields in the IP header itself. On the other hand, ESP provides for encryption and/or integrity protection only for the payload[5] in the packet. Accordingly, as seen in Figure 3.7, the AH contains only the integrity check of the packet whereas the ESP fields support both the IV (used for encryption) and the integrity check of the packet. Most of the other fields like the SPI, sequence number and payload length are common between the two headers.

Now that we know what IPSec headers look like, the next question is who inserts them? The obvious answer is—the system implementing IPSec. However, this requires a little more explanation. IPSec may be used by communication end points

[4] This may happen, for example, if multiple processes at a single IP address (computer) are talking to different IP addresses (computers).

[5] And not for fields in the IP header.

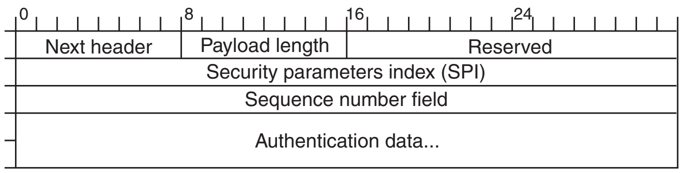


Figure 3.7a: Authentication Header for IPSec

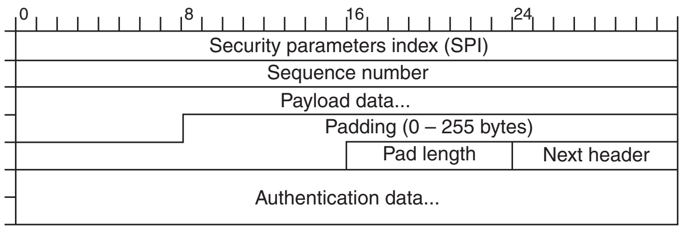


Figure 3.7b: ESP Header for IPSec

but also by intranet gateways to provide a secure "tunnel" between two intranets (secure environment) over the Internet (unsecure environment). In the former case, the IPSec headers are inserted between the IP header and the packet payload/data.

In the tunneling case, however, the IPSec header is inserted around the IP packet and another new IP header is inserted around this encapsulated packet. The new IP header

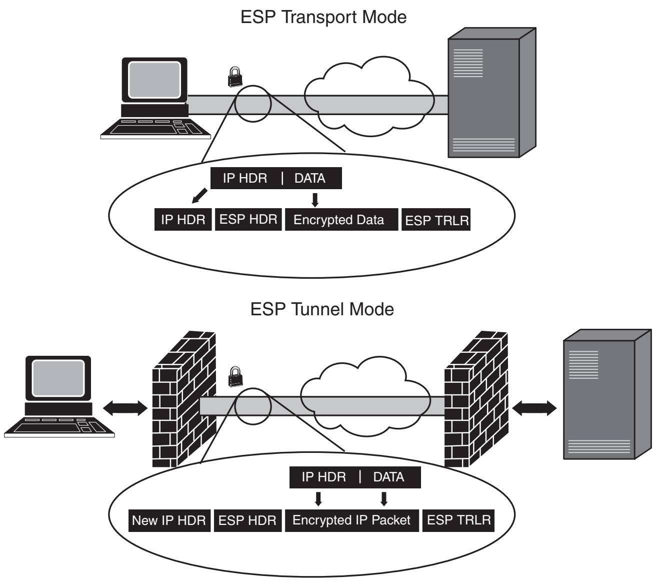


Figure 3.8a: ESP Modes of Operation

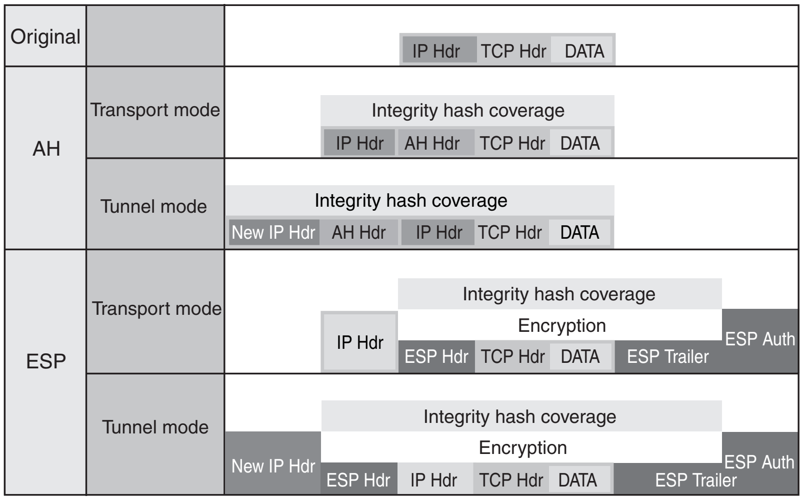


Figure 3.8b: Encapsulation in Different IPSec Modes

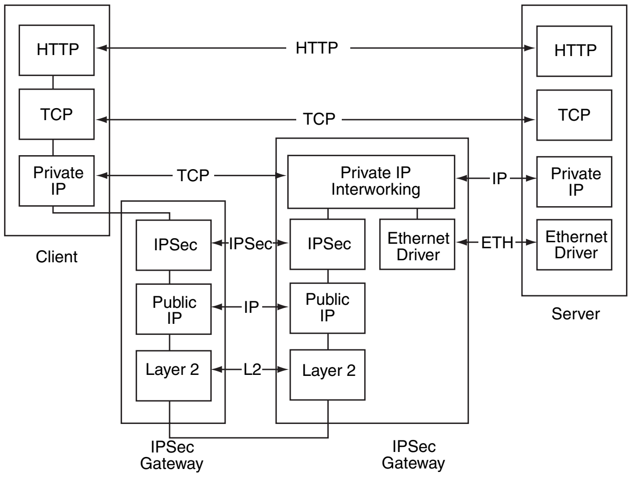


Figure 3.8c: IPSec in the Layered Model

contains the IP addresses of the endpoints of the tunnel, since it is at the remote gateway that this external IP header and the IPSec header are removed to restore the old packet. Even though the tunneling encapsulation is costly in terms of overhead incurred, it is a very attractive proposition for corporate solutions which wish to connect intranets without having to install IPSec at each and every client in the network.

Now that we have learnt about the IPSec headers which provide for encryption and integrity protection, we will look at the other part of IPSec: IKE. IKE is a protocol used for mutual authentication and SA establishment. Remember that the SA includes the session keys to be used for authentication and/or encryption. IKE is a complex protocol made even more complex by the numerous variations in which it can be used.

The IKE variations stem from the various key types supported by IPSec. There are three types of keys supported: preshared secret key (think SKC), public encryption key (and a private decryption key), and public signature key (and a private verification key). Furthermore, each key can be used in one of two modes: main mode and aggressive mode. The former offers more security features like identity protection and secure negotiation of cryptographic parameters whereas the latter is more efficient in terms of the numbers of messages exchanged (and therefore the bandwidth and time consumed). Now, we already have six variations of IKE and that's not the end of the story. There are two more modes, since there are two variations of using the public encryption key: original and revised. So, there you have it: eight variations of IKE.

Why are there so many modes? It seems that the IPSec designers wanted to be flexible and support as many key types in as many situations as possible. Certainly a noble aim, but IPSec is a perfect example of what happens when you aim too high. The IPSec protocol specification is so huge that its size serves as a deterrent for most people to use or implement it, let alone analyze it. The lesson from the IPSec story (probably) is to keep it simple. Since the protocol is huge, it is impractical to describe each and every mode in detail. Instead, we look at the underlying conceptual protocol design.

IKE consists of two phases. Phase 1 does mutual authentication and establishes the session keys. Using the session keys established in Phase 1, multiple Phase 2 SAs may be established. To understand the motivation behind splitting the protocol into two phases, realize that Phase 1 is usually based on PKC, whereas Phase 2 may be based on SKC without compromising security. Such a scenario might occur, for example, if IPSec is being used to setup a Virtual Private Network (VPN) between two LANs. This would require the creation of a firewall-to-firewall link going over the public Internet. Since traffic from multiple users would be going over the IPSec link, it is more efficient to have the firewalls and gateways mutually authenticate each other using Phase 1 of IKE. The session keys resulting from this Phase 1 may then be used to establish a SA in Phase 2 on a per flow[6] basis.

[6] A flow refers to an end-to-end/user-to-user session.

Figure 3.9 shows how Phase 1 of IKE proceeds in the main mode. Messages 1 and 2 are used for negotiating the cryptographic parameters to be used for IPSec. These cryptographic parameters include the encryption algorithm (DES, 3DES and so on), the hash algorithm (MD5, SHA, and so on), authentication method (preshared keys, public signature key, public encryption key, and so on) and the Diffie-Hellman group (modular exponentiation, elliptic curve) and so on. Messages 3 and 4 carry out the Diffie-Hellman exchange. At the end of messages 3 and 4, Alice and Bob have established a shared secret. This shared secret is then used to protect messages 5 and 6. Messages 5 and 6 are used by each user to reveal and prove her identity.

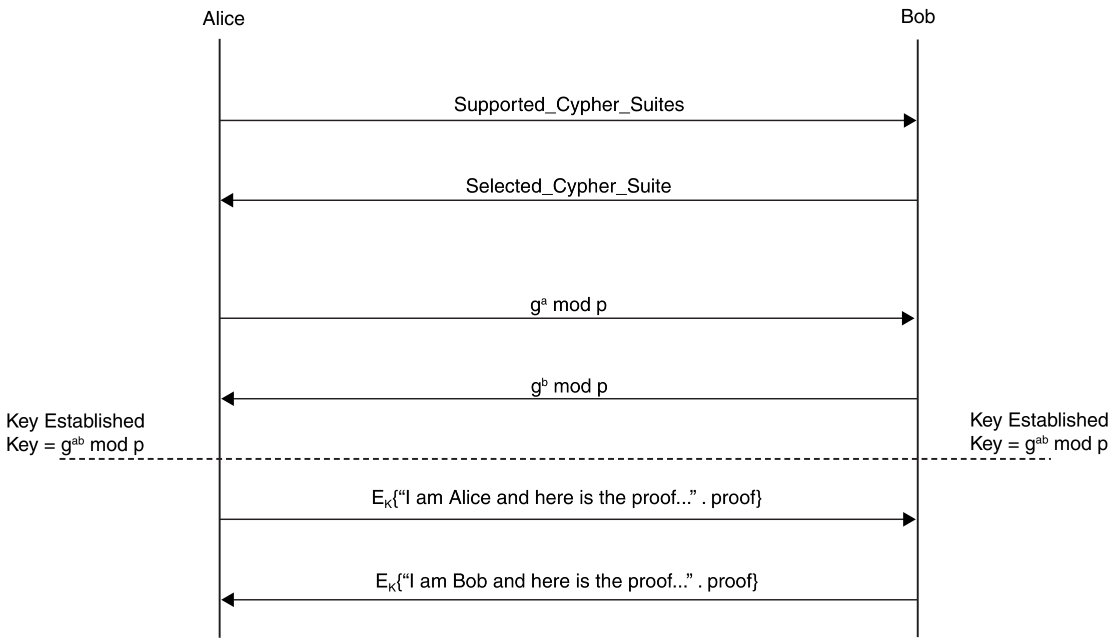


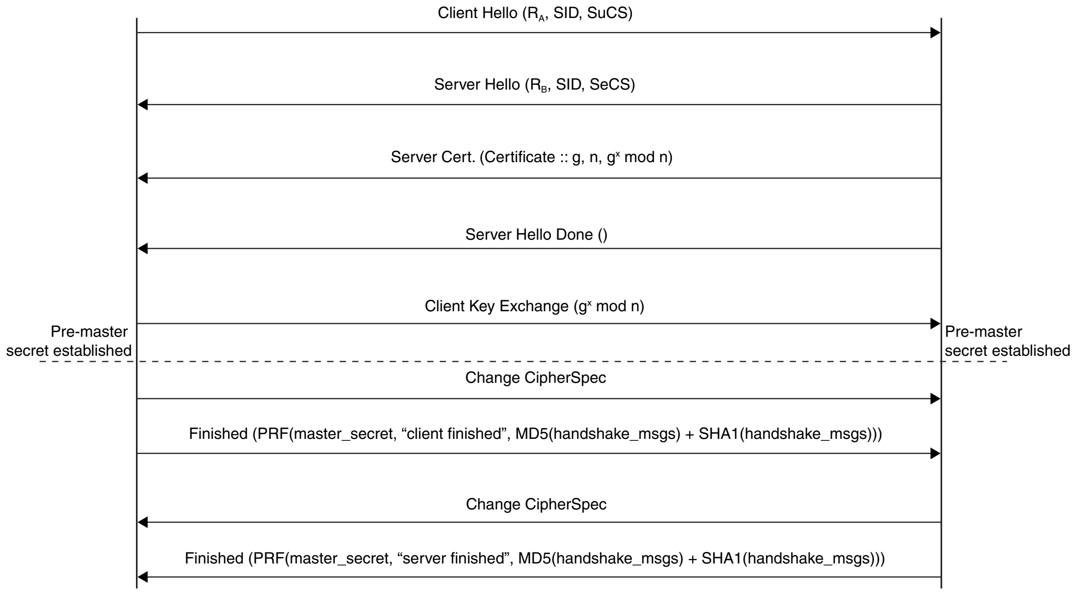
Figure 3.9: IKE overview

3.5 Security at Layer 4: SSL/TLS

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are slightly incompatible protocols but are similar enough to be described together. The SSL/TLS protocol is based on the philosophy that it is easier to modify the application than the operating system. Therefore, SSL sits between the application layer and the transport layer. Applications interface to the SSL layer rather than the transport layer. From a programming viewpoint, applications use SSL sockets rather than transport layer sockets for network programming.

The name of the protocol is therefore self-explanatory. SSL is a layer (a library of wrapper functions) around the socket interface. It runs as a user process on top of Transmission Control Protocol (TCP) (port number 443). Applications which wish to use SSL/TLS should use the SSL library functions for networking instead of directly calling the socket functions. This allows the two communicating parties to authenticate and establish a session key that is used to protect the remainder of the session cryptographically. The obvious assumption here is that both the communicating applications must be using SSL/TLS.

Figure 3.10 shows how the client, Alice, authenticates with the server Bob and establishes a secret (referred to as the premaster secret). The TLS handshake begins with Alice sending a *Client-Hello* message to Bob, though this may in some cases be preceded by the server (Bob) prompting the client (Alice) to send a *Client-Hello* message using the *Hello-Request* message. The *Client-Hello* message sent by Alice contains, among other things, a random number generated by Alice, the Session-Id and a list of cipher-suites supported by Alice. The Session-Id is Null for a new connection but may be used by Alice for session resumption in the near future.



Client Hello (R_A, SID, SuCS)
Server Hello (R_B, SID, SeCS)
Server Cert. (Certificate :: g, n, g^x mod n)
Server Hello Done ()
Client Key Exchange (g^x mod n)
Pre-master secret established
Pre-master secret established
Change CipherSpec
Finished (PRF(master_secret, "client finished", MD5(handshake_msgs) + SHA1(handshake_msgs)))
Change CipherSpec
Finished (PRF(master_secret, "server finished", MD5(handshake_msgs) + SHA1(handshake_msgs)))

Figure 3.10: TLS

On receiving the *Client-Hello* message from Alice, Bob responds with a *Server-Hello* message which contains, among other things, a random number generated by Bob, the Session-Id and a selection from the list of cipher-suites that Alice supplied in the *Client-Hello* message. If the Session-Id in the Client-Hello message was Null, that is, if this is a new connection, Bob generates a Session-Id and returns it in this message. If, on the other hand, this is a session-resumption attempt from Alice, the Session-Id would be non-Null. In this case, Bob would look up its session cache to find this Session-Id. If a match is found and Bob is willing to establish the new connection using the previously established parameters, Bob would send back a Finished message with the same Session-Id that it received in the *Client-Hello* message. If Bob does not find a matching Session-Id in its cache or if Bob is not willing to resume the session, Bob will send back a *Server-Hello* message with a different Session-Id.

The *Sever-Hello* message from Bob is followed by a *Server-Certificate* message which contains a certificate type appropriate to the selected cipher-suites' key-exchange algorithm. Figure 3.10 assumes that the DH key exchange algorithm is in use. In this case, the *Server-Certificate* is followed by a *Server-Hello-Done* message. Note that some key-exchange algorithms (like RSA) may require an additional *Server-Key-Exchange* message to be sent before the *Server-Hello-Done* message.

On receiving the *Server-Hello-Done* message from Bob, Alice sends the *Client-Key-Exchange* message to Bob. In the DH case, this message contains Alice's public key. After this message, both Alice and Bob can derive the premaster secret independently without worrying about Eve possessing this secret. For why this is so, see Sections 2.2.1 and 2.2.2. The premaster secret is then converted to a master secret using the pseudo-random-function, the client random number and the server random number; that is:

master_secret = PRF(pre_master_secret, "master secret", ClientHello.random+ServerHello.random)

where "+" denotes a concatenation. The pseudorandom-function for TLS version 1.0 is created by splitting the premaster secret into two halves and using one half to generate data with the MD5 MAC and the other half to generate data with the SHA1 MAC.

The *Client-Key-Exchange* message from Alice is followed by the *Change-Cipher-Spec* message which indicates to Bob that from now on Alice is going to use the derived master-secret to protect the rest of the session. The first message protected with the just-negotiated algorithms, keys and secrets that Alice sends to Bob is the *Finished* message. The *Finished* message contains as data the signed hash of all the handshake messages exchanged between Alice and Bob up until now. This means that the recipient of the *Finished* message can use the data contained in this message to verify that a) the other end has derived the master secret correctly and b) none of the handshake messages have been modified or spoofed by a malicious Eve. On receiving the *Finished* message from Alice, Bob also sends a *Change-Cipher-Spec* message followed by a *Finished* message. From now on, applications at both ends can start sending data securely.

To summarize, TLS uses the Hello messages to agree on cipher-suites (algorithms), exchange random numbers and check for session resumption. Then, the client and the server exchange certificates and cryptographic parameters to authenticate themselves and "independently" derive a premaster secret. Using this premaster secret and the random values exchanged earlier, both endpoints derive a master secret which is used as a basis for protecting the rest of the session.

There are two important things to note about the Session-Id used in TLS. One, that the Session-Id becomes valid only when the handshake negotiating it completes with the exchange of the Finished message and persists until it is removed due to aging or because of a session error. Two, since the contents of the handshake as a whole are protected by the *Finished* messages exchanged at the end of the handshake, the Session-Id cannot be spoofed by a malicious Eve.

Perhaps the most important and known security loophole of SSL is the DoS attack against SSL. Since SSL runs on top of TCP, it relies on TCP for communication with the remote end. TCP packets check against transmission errors using a checksum. However, this checksum is not cryptographically protected.[7] Now, consider a scenario where Mallory inserts malicious data packets into a packet stream which is protected by SSL. If Mallory can ensure that the data packets she inserts would pass the TCP checksum[8] at the receiver, the TCP layer at the receiver has no way of knowing that this TCP packet is invalid. The TCP layer at the receiver would therefore pass this packet up to the SSL layer. SSL will correctly drop the packet since it would fail the integrity check of SSL. No harm done—right?

Well, not exactly. The problem is that SSL has no way of telling TCP that the data packet that it just handed over was bogus. Why? The layered architecture is based on a service model where each layer provides a set of services to the layer above it and the interaction between the layers is minimized in order to be modular; in other words, each layer should function as independently as possible. One implication of this architecture is that data stream handling at each layer should be independent. For our case, this means that TCP should independently determine the validity of each packet without interference from other layers. Hence there is no provision in the API to TCP to let higher layers tell TCP that the packet was bogus. The TCP/IP architecture was based on the theory that if the higher layer determines that a packet

[7] The TCP checksum was designed to protect against transmission errors and not as a security mechanism.
[8] Not very difficult to do since the checksum is not cryptographically protected.

is invalid, it would simply drop the packet. This is a typical example of what happens when security is added as an afterthought and not as an integral part of the system architecture.

Anyway, continuing with our example, what happens is that TCP is unaware of the fact that it has delivered a bogus packet. Now, when the real packet arrives, TCP discards it, since it will have the same sequence number as the bogus packet and TCP would determine this as a duplicate packet. Meanwhile, SSL is missing a packet it is expecting. This would result in SSL eventually closing the connection after a timeout. So, what Mallory has achieved is the closing of the SSL connection; in other words a denial of service attack.

3.6 Security at Layer 5+

The OSI stack was an architectural framework meant to serve as a guideline for designing network protocols. The most widely deployed protocol today is IP. IP is a Layer 3 protocol which can be implemented on top of any Layer 2 protocol. The primary protocols used with IP at Layer 4 are TCP and UDP.

For our purposes, the thing to note is that the OSI Layers 5 and above are usually collapsed into one single layer and implemented in the application itself. It is for this reason that this section clubs together Layers 5 and above.

Because of the multitude of applications which use networks today, it is impractical to describe the security implementation at this layer. Each application has a unique set of security requirements and it is up to the application designers to implement security.

CHAPTER 4

Voice-Oriented Wireless Networks

In this chapter, we take a break from studying security and instead study the wireless medium. What is it that makes a wireless medium so unique? What are the problems of operating in the wireless medium and how are they overcome? What are the different types of wireless networks in use today? How does each one of them work and how do they differ from each other? The aim of this chapter is to answer these questions so as to establish a context in which wireless security can be studied in the following chapters.

The first successful commercial use of wireless telecommunication was the deployment of cellular phones (mobile phones). In this book, we refer to these networks as Traditional Wireless Networks (TWNs). These networks were designed with the aim of extending the existing wired Public Switched Telephone Network (PSTN) to include a large number of mobile nodes. The deployment of TWNs allowed users to be mobile and still make voice calls to any (fixed or mobile) phone in the world. In other words, TWNs were designed as a Wide Area Network (WAN) technology enabling voice communication. These networks have evolved over time to support both voice and data communication but the underlying feature of the TWNs being a WAN technology is still true.

For a long time, TWNs were the predominant example of wireless telecommunication. In the late 1990s, another wireless technology emerged: wireless local area networks (WLANs). Unlike TWNs, WLANs were designed primarily with the aim of enabling data communication in a limited geographical area (local area network). Though this aim may seem counter-intuitive at first (why limit the geographical coverage of a network?), this principle becomes easier to understand when we think of WLANs as a wireless Ethernet technology. Just as Ethernet (IEEE 802.3) provides the backbone of wired Local Area Networks (LANs) today, IEEE 802.11 provides the backbone of wireless LANs. Chapter 5 studies WLANs.

Just as TWNs were initially designed for voice and over time evolved to support data, WLANs were initially designed for data and are now evolving to support voice.

Probably the most prominent difference between the two standards is that the former is a WAN technology and the latter is a LAN technology. TWNs and WLANs are today the two most dominant wireless telecommunication technologies in the world. Analysts are predicting the convergence (and co-existence) of the two networks in the near future.

Finally, we are today seeing the emergence of wireless Mobile Ad Hoc Networks (MANETs). Even though this technology is still in its research phase, it promises to have significant commercial applications in the near future. As the name suggests, MANETs are designed with the aim of providing ad hoc communication. Such networks are characterized by the absence of any infrastructure and are formed on an as-needed (ad hoc) basis when wireless nodes come together within each others' radio transmission range. We look at the underlying concepts of MANETs in Section 5.2.

We begin however by looking at some of the challenges of the wireless medium.

4.1 The Wireless Medium

4.1.1 Radio Propagation Effects

The wireless medium is a harsh medium for signal propagation. Signals undergo a variety of alterations as they traverse the wireless medium. Some of these changes are due to the distance between the transmitter and the receiver, others are due to the physical environment of the propagation path and yet others are due to the relative movement between the transmitter and the receiver. We look at some of the most important effects in this section.

Attenuation refers to the drop in signal strength as the signal propagates in any medium. All electromagnetic waves suffer from attenuation. For radio waves, if *r* is

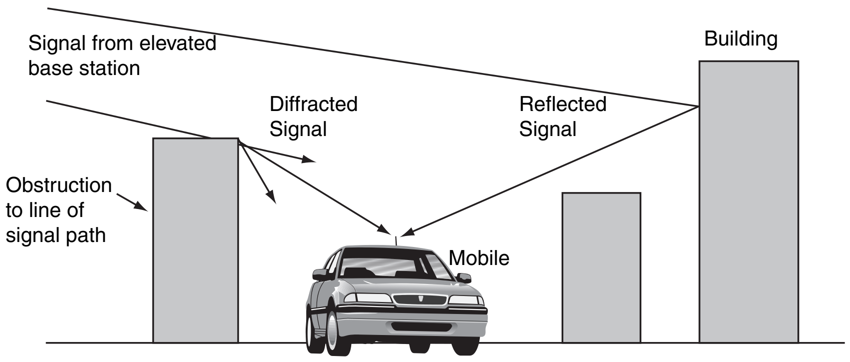

Signal from elevated base station
Building
Diffracted Signal
Reflected Signal
Obstruction to line of signal path
Mobile

Figure 4.1a: Signal Propagation in Wireless Environment

the distance of the receiver from the transmitter, the signal attenuation is typically modeled as $1/r^2$ at short distances and $1/r^4$ at longer distances; in other words, the strength of the signal decreases as the square of the distance from the transmitter when the receiver is "near" the transmitter and as the fourth power when the receiver is "far away" from the transmitter. The threshold value of r where distances go from being "near" to being "far away" are referred to as the reference distance. It is important to emphasize that this is radio modeling we are talking about. Such models are used for simulation and analysis. Real-life radio propagation is much harsher and the signal strength and quality at any given point depends on a lot of other factors too.

Attenuation of signal strength predicts the average signal strength at a given distance from the transmitter. However, the instantaneous signal strength at a given distance has to take into account many other effects. One of the most important considerations which determine the instantaneous signal strength is, not surprisingly, the operating environment. For example, rural areas with smooth and uniform terrain are much more conductive to radio waves than the much more uneven (think tall buildings) and varying (moving automobiles, people and so on) urban environment. The effect of the operating environment on radio propagation is referred to as shadow fading (slow fading). The term refers to changes in the signal strength occurring due to changes in the operating environment. As an example, consider a receiver operating in an urban environment. The path from the transmitter to the sender may change drastically as the receiver moves over a range of tens of meters. This can happen if, for example, the receiver's movement resulted in the removal (or introduction) of an obstruction (a tall building perhaps) in the path between the transmitter and the receiver. Shadow fading causes the instantaneous received signal strength to be lesser than (or greater than) the average received signal strength.

Another propagation effect that strongly effects radio propagation is Raleigh fading (fast fading). Unlike slow fading which effects radio propagation when the distance between the transmitter and the receiver changes of the order of tens of meters, fast fading describes the changes in signal strength due to the relative motion of the order of a few centimeters. To understand how such a small change in the relative distance may affect the quality of the signal, realize that radio waves (like other waves) undergo wave phenomena like diffraction and interference. In an urban environment like the one shown in Figure 4.1a, these phenomena lead to multipath effects; in other words, a signal from the transmitter may reach the receiver from multiple paths. These multiple signals then interfere with each other at the receiver. Since this interference can be either constructive or destructive, these signals may either reinforce each other or cancel each other out. Whether the interference is constructive or destructive depends on the path length (length the signal has traveled) and a small change in the path length can change the interference from a constructive to a destructive one (or vice versa). Thus, if either of the transmitter or the receiver move even a few centimeters, relative to each other, this changes the interference pattern of the various waves arriving at the receiver from different paths. This means that a constructive interference pattern may be replaced by a destructive one (or vice versa) if the receiver moves by as much as a few centimeters. This fading is a severe challenge in the wireless medium since it implies that even when the average signal strength at the receiver is high there are instances when the signal strength may drop dramatically.

Another effect of multipath is inter-symbol interference. Since the multiple paths that the signal takes between the transmitter and the receiver have different path lengths, this means that the arrival times between the multiple signals traveling on the multiple paths can be of the order of tens of microseconds. If the path difference exceeds 1 bit (symbol) period, symbols may interfere with each other and this can result in severe distortion of the received signal.

4.1.2 Hidden Terminal Problem

Wireless is a medium which must be shared by all terminals which wish to use it in a given geographical region. Also, wireless is inherently a broadcast medium since radio transmission cannot be "contained." These two factors create what is famously known as the *hidden terminal problem* in the wireless medium. Figure 4.1b demonstrates this problem.

Figure 4.1b shows three wireless terminals: A, B and C. The radio transmission range of each terminal is shown by a circle around the terminal. As is clear, terminal B lies within the radio transmission range of both terminals A and C. Consider now what happens if both A and C want to communicate with B. Most media access rules for a shared medium require that before starting transmission, a terminal "senses" the

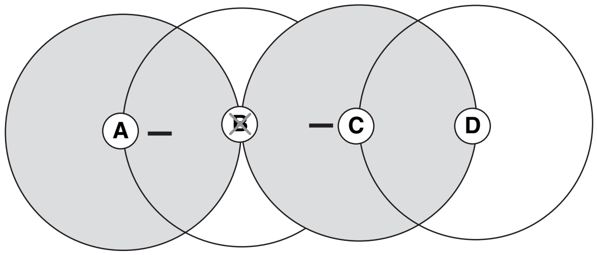


Figure 4.1b: Hidden Node Problem

medium to ensure that the medium is idle and therefore available for transmission. In our case, assume that A is already transmitting data to B. Now, C also wishes to send data to B. Before beginning transmission, it senses the medium and finds it idle since it is beyond the transmission range of A. It therefore begins transmission to B, thus leading to collision with A's transmission when the signals reach B. This problem is known as the hidden terminal problem since, in effect, A and C are hidden from each other in terms of radio detection range.

4.1.3 Exposed Terminal Problem

The exposed terminal problem is at the opposite end of the spectrum from the hidden terminal problem. To understand this problem, consider the four nodes in Figure 4.1c.

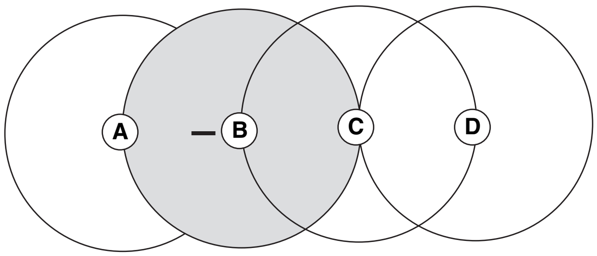


Figure 4.1c: Exposed Node Problem

In this example, consider what happens when B wants to send data to A and C wants to send data to D. As is obvious, both communications can go on simultaneously since they do not interfere with each other. However, the carrier sensing mechanism raises a false alarm in this case. Suppose B is already sending data to A. If C wishes to start sending data to D, before beginning it senses the medium and finds it busy (due to B's ongoing transmission). Therefore C delays its transmission unnecessarily. This is the exposed terminal problem.

4.1.4 Bandwidth

"The Queen is dead.
Long Live the Queen."

Bandwidth is one of the most important and one of the most confusing topics in telecommunications today. If you keep update with the telecommunication news, you would have come across conflicting reports regarding bandwidth. There are a lot of people claiming "bandwidth is cheap" and probably as many people claiming "it is extremely important to conserve bandwidth." So, what's the deal? Do networks today have enough bandwidth or not?

The problem is there is no single correct answer to that. The answer depends on where you are in the network. Consider the core of the IP and the PSTN networks: the two most widely deployed networks today. The bandwidth available at the core of these networks is much more than required: bandwidth therefore is cheap at the core of the network. Similarly the dawn of 100 Mbps and Gigabit Ethernet has made bandwidth cheap even in the access network (the part of the network that connects the end-user to the core). The wireless medium however is a little different and follows a simple rule: bandwidth is always expensive. This stems from the fact that in almost all countries the wireless spectrum is controlled by the government. Only certain bands of this spectrum are allowed for commercial use, thus making bandwidth costly in the wireless world. All protocols designed for the wireless medium therefore revolve around this central constraint.

4.1.5 Other Constraints

Bandwidth is not the only constraint for wireless networks. One of the prominent "features" of wireless networks is that they allow the end user (and hence the end node or terminal) to be mobile. This usually means that the devices that the wireless nodes which are being used need to be small enough to be mobile. This in turn means additional constraints: small devices mean limited processing power and limited battery life. Also small nodes means that they are easy to lose or steal, thus having security implications.

4.2 The Cellular Architecture

The concept of cellular architecture is a great example of how engineering adapts an obstacle into an opportunity. Recall from Section 4.1.1, that radio waves attenuate significantly with distance: this limits the transmission range of the medium. Similarly, another important challenge in the wireless medium is the limited bandwidth that is available. This limited bandwidth means that it is a challenge for service providers to support enough simultaneous voice calls to justify the deployment costs of a wireless network.

What the cellular concept does is to solve each of these challenges by exploiting the other "challenge." To understand the cellular concept, we divide the geographical area that the wireless network needs to cover into hexagonal cells. Figure 4.2 shows how a geographical area can be divided into hexagonal cells.

The cellular architecture exploits the fact that the signal strength in the wireless world decays as the square of the distance from the transmitter (as a factor of three or four in urban environments). This means that a single carrier frequency can be used in

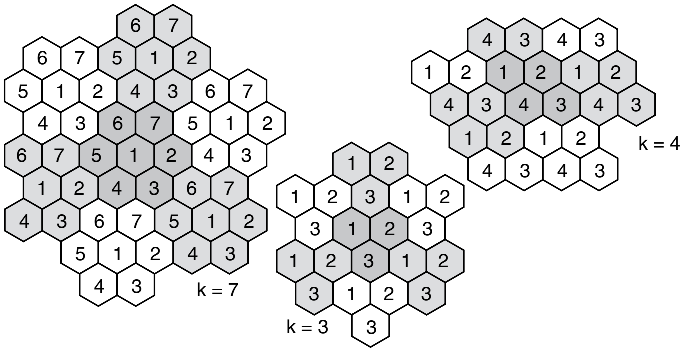


Figure 4.2: Cellular Architecture

multiple geographical locations provided that these locations are geographically well separated from each other. In other words, if two "cells" are far enough from each other to avoid interference, they can use the same part of the wireless spectrum, thus making wireless a viable commercial medium. In Figure 4.2, we see three examples of the cellular architecture with different frequency reuse factors. The term frequency reuse factor (k) defines the number of cells in a cluster and therefore determines how far the cells reusing the same frequency are from each other. In Figure 4.2, we show three most often used values of k. The cells labeled with the same number use the same frequency.

Figure 4.3 looks at detail in how frequencies are reused in a cellular network. A set of neighboring cells forms a cluster. Within a cluster, each cell uses a unique frequency. However, cells outside the cluster may re-use the frequencies used in the cells. Network planning for cellular architecture involves, among other things, allocating frequencies to be used in each cell so as to maximize the distance between cells using the same frequency (referred to as co-channel cells). For the hexagonal cell model,

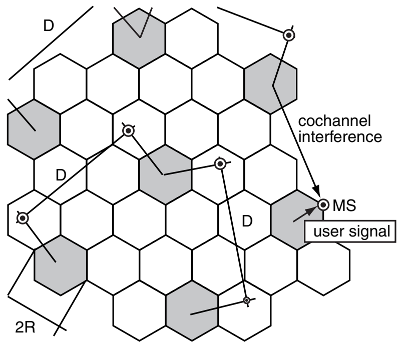


Figure 4.3: Cellular Architecture—*Details*

the value of k should be of the form $i^2 + j^2 + i.j$ where i and j must be integers. In this model, the co-channel cell is located i cells along adjacent cells and j cells along the 60 degree mark.

To understand how the frequency reuse of cellular architecture increases system capacity, let us consider the following scenario:

B MHz: Bandwidth available to be used.
S: Number of duplex channels that can be deployed in B MHz.

Without the cellular architecture, the system capacity for this network would be *S*. Now, what happens if we use the cellular architecture and divide this network as follows:

k: Frequency re-use factor; that is, the number of cells which form a cluster.
M: Number of clusters that are needed to cover the geographical area under coverage.

Now, since each cluster can use the *S* duplex channels, the system capacity has increased M-fold. The next question is pretty obvious. What prevents us from increasing *M* infinitely and making the system have infinite capacity? The answer is that given a fixed geographical area, increasing *M* means reducing *k* and reducing *k* means that the cells using the same frequency come closer to each other. Obviously, this is not good. The whole concept of cellular architecture is to have cells using the same frequency far apart geographically. By reducing *k*, we bring these cells closer to each other. This leads to an increase in the co-channel interference (i.e., the interference among two cells using the same frequency). To understand this tradeoff, we define the co-channel re-use ratio (*Q*) as Q = D/R where *R* is the "radius" (distance from the center to a of the hexagonal cell to a vertex) and *D* is the distance between cells using the same frequency.

Using hexagonal geometry, $D/R = (3k)^{1/2}$

Now, by definition Signal to Interference Ratio (SIR) = (desired signal strength)/(sum of all interference)

Since signal strength decays with distance as a power of n,

$SIR = R^{-n}/\sum D^{-n}$

For a symmetrical hexagonal model, assuming first-tier interference only,

$SIR = 1/6\ (D/R)^n = 1/6\ (3k)^{n/2}$

It is this SIR that prevents the service provider from reducing *k* indefinitely to achieve higher and higher system capacity. A reduction in *k* means a reduction in the signal to interference ratio and therefore a deterioration of voice quality for the user. Therefore the value of k is an engineering tradeoff to balance the SIR with the system capacity.

4.3 TWNs: First Generation

The earliest wireless cellular communication systems were deployed in 1980 and 1981 in Japan and Scandinavia. In the following years various cellular systems were developed and deployed all over the world. Together these came to be known as the first generation cellular systems. Even though these standards were mutually incompatible, they shared many common characteristics. The most prominent among them was that voice transmission was done by means of frequency modulation; that is, the air-interface in these standards was analog.

One of the most well documented first generation wireless cellular systems was the Advanced Mobile Phone System (AMPS) in North America. Figure 4.4a shows the prominent network components in the AMPS architecture. The Mobile Station (MS) is the end-user terminal that communicates over the wireless medium with the Land Station (LS). The land station (also known as base transceiver station) is connected by land lines[1] to the Mobile Telephone Switching Office (MTSO). This was the AMPS

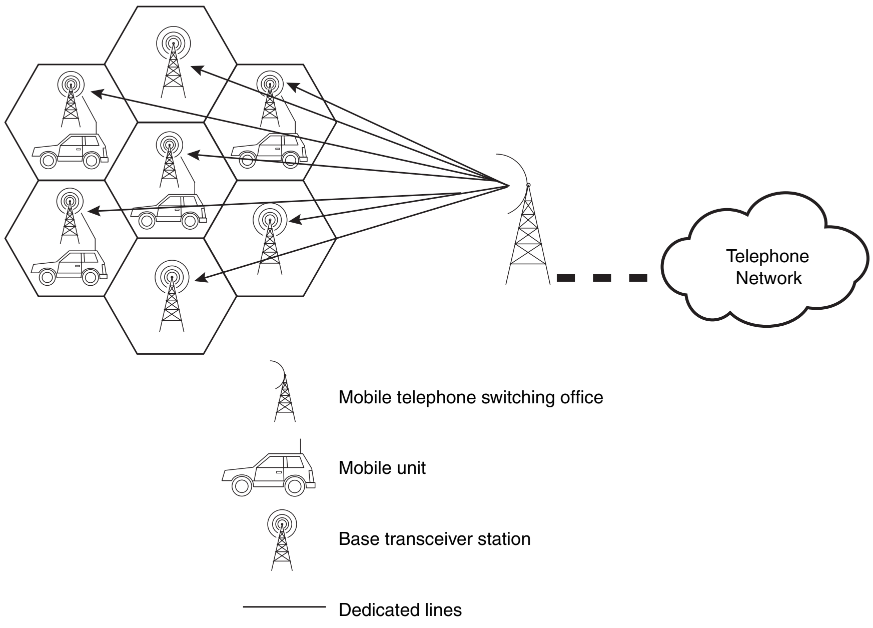


Figure 4.4a: AMPS Layout

[1] Land lines may physically be copper wires, optical fibers or microwave links.

architecture. The deployment of an AMPS wireless network required the deployment of MTSOs, LSs and the end-user mobile stations.

Before we go ahead, it is instructive to take a step back and realize what the purpose of TWNs was. The aim of TWNs was to extend the PSTN. In other words, the aim of the AMPS network was to allow a MS to make telephone calls to any other phone in the world. How is this achieved? To answer this question, we first look at how telephone calls work in the wired PSTN.

The PSTN really consists of two logically separate networks: the signaling network and the media network. To understand the difference between signaling and media, consider what happens when you pick up your telephone and make a call. Signaling refers to the overall process of going off hook, getting a dial tone, dialing digits, getting a ring back and finally getting a call connected. The media network comes into play only after the call is connected and is used for carrying the voice. These two logically separate networks are implemented as two physically separate networks in the PSTN as shown in Figure 4.4b.

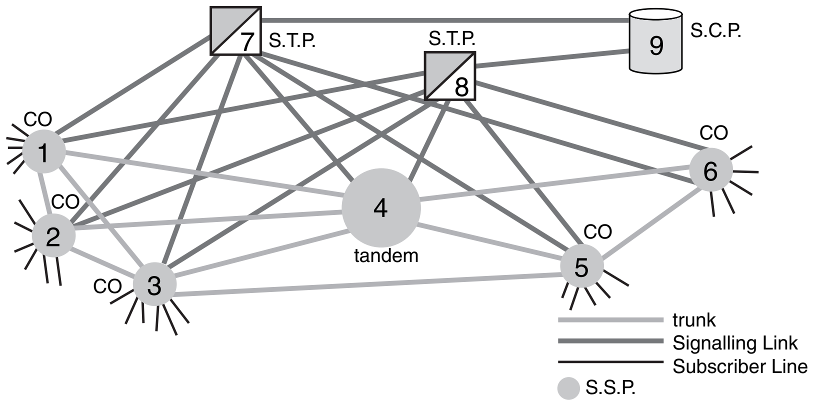


Figure 4.4b: SS7 and the PSTN

The media network consists of the physical wires (trunks) which carry voice calls and the switches which connect these trunks. It is the media network which reaches the end users at home. The end user's phone is connected to the local connection office also known as the local telephone exchange Central Office (CO). These local telephone exchanges are connected to each other and to the tandem office by trunks. The trunks are used for carrying voice traffic between the switches. The media network is therefore responsible for carrying voice traffic from one end-user to another. In Figure 4.4b, entities labeled 1 through 6 and the trunks connecting them form the media network.

The signaling network, on the other hand, is responsible for call control (call management) which includes call setup, call routing, call maintenance and call termination. The signaling network in the PSTN uses Signaling System #7 (SS7) for call control. SS7 is an Out-Of-Band (OOB) Common Channel Signaling (CCS) system. This means that the SS7 messages are carried on a logically separate network (out-of-band)[2] than the voice calls and that the signaling messages for all voice calls use this same network (common-channel). The SS7 network basically consists of Signaling Points (SP) exchanging control messages to perform call management.[3] There are primarily two types of signaling points: SSP and STP.

A Service Switching Point (SSP) is a SP which is co-resident with the local connection office (or the tandem office) and has the ability to control the voice circuits of its switch. The SSP is a logical entity which may be implemented in the switch itself or it may be a physically separate computer connected to (and controlling) the switch. In Figure 4.4b, entities labeled 1 through 6 would each have a SSP associated with them. Note that for the purpose of the signaling network, these entities are not connected to each other: the trunks which connect them are part of the media network which is used for carrying voice, not signaling messages. A Signaling Transfer Point (STP) is a SP which is capable of routing call control messages between the SSPs. An STP is therefore used by one SSP to route messages to another SSP. In Figure 4.4b, entities labeled 7 and 8 are STPs.

To understand the overall picture, realize that since the PSTN media network is a connection-oriented network, the end-to-end connection between the calling party and the called party needs to be established before the call is "connected." This means that all switches in the media-path need to reserve resources (bandwidth, buffers and so on) as part of signaling. As an example, we go back to what happens when you pick up your phone and make a call. Suppose that you are connected to CO3 in Figure 4.4b. The dialed digits reach CO3 which analyzes these digits to determine that the called party telephone number is connected to CO5. The SSP at CO3 therefore sends a call setup message to the SSP at CO5 via STP8. On receiving this message the SSP at CO5 reserves the resources required for this call and sends back an acknowledgment message to CO3 specifying the trunk selected for carrying the voice call. When CO3 receives this acknowledgment, it connects you to your called number and the connection is established.

[2] The nomenclature makes sense if you see the media network as the band carrying the voice.

[3] SS7 also specifies other nodes like an SCP used for advanced services, but those are irrelevant for the purposes of this discussion.

Now let us look at what happens in the AMPS network. When the mobile user dials a phone number, this phone number is relayed from the LS to the MTSO. The MTSO is basically a CO enhanced to support mobility in the wireless medium. Just like its wired counterpart, the MTSO consists of a switch connected to the media network of the PSTN and a SSP connected to the SS7 network. When the MTSO gets the called-party number, it uses the same procedure as any another CO to route the call. The PSTN is not aware that the end-user is a wireless user and it sees the MTSO as just another CO. This makes routing calls between the MTSO and the PSTN easy. But what happens after the call reaches the MTSO? How does it get from the MTSO to the end-user's phone? The MTSO is responsible for taking care of MS mobility; that is, it is up to the MTSO to find or know the location of a MS at any given time. This responsibility of the MTSO is known as location management. We see how this is accomplished in Section 4.3.2 but before that, we need to understand the various addresses used in the AMPS architecture.

4.3.1 Addresses in AMPS

The Mobile Identification Number (MIN) is the 34-bit (10-digit in the USA) telephone number assigned by the service provider (operating company) to the subscriber. The MIN uniquely identifies the subscriber.

The Electronic Serial Number (ESN) is a 32-bit number assigned permanently to the MS by the manufacturer before shipping. The ESN uniquely identifies the physical equipment (phone).

The Station Class Mark (SCM) is a 4-bit identification code stored in the MS, which describes the capabilities of the MS.

The System Identifier (SID) is a 15-bit code which identifies the service provider in a specific geographical area. It is stored by each LS of the service provider. Each MS also stores the SID of its service provider. In effect, the MS compares its stored SID with the SID of the LS it is connected to, to determine whether it is in its home network or is roaming.

The Supervisory Audio Tone (SAT) and the Digital Color Code (DCC) are used to distinguish between the various LSs of a service provider.

4.3.2 Call Setup in AMPS

A call originating from the PSTN destined to a MS in an AMPS network proceeds as follows:

1. Call-setup messages reach MTSO through the PSTN.
2. The MTSO sends a ringing tone to the calling party.
3. In order to locate the MS, the MTSO sends a Page command (including the MIN of the called party) to all the land stations (LSs) that it controls.
4. Each LS receiving the Page message broadcasts the message in its cell.
5. The terminating MS sends a Page-response to the LS. The page-response contains both the MIN and the ESN.
6. The LS relays this information to the MTSO using a service request.
7. The MTSO authenticates the MS by comparing the received ESN with the ESN stored in the subscriber's home MTSO database.
8. After successful authentication, the MTSO instructs the LS to assign a voice channel. This includes selecting the uplink and the downlink frequency bands to be used for carrying voice for this call.
9. The LS informs the MS of the physical channel frequencies to use for the voice channel.
10. At this point, the call is established and voice begins to flow.

A call originating from the MS destined to a phone in the PSTN proceeds as follows:[4]

1. When the user presses the "send" button on their phone, the MS sends an Originate (call request) containing its MIN, ESN, SCM and the called party number to the LS.
2. The LS relays this message to the MTSO.
3. MTSO does authorization, validation and connection setup over the PSTN. The MTSO then tells the LS to allocate voice channels to the MS for this call.
4. The LS allocates voice channels for the MS and informs the MS about these.
5. The MS can now start the voice conversation using the voice channels allocated to it.

The first generation cellular networks fulfilled the original goals that they were designed to achieve. They provided wide area wireless coverage with low probabilities of call blocking and call dropping, high transmission quality, high user

[4] We assume here that the MS is already "associated" with an LS in the AMPS network.

mobility, high spectrum efficiency and early deployment. Even though these first generation cellular networks were undoubtedly highly successful, they had a number of drawbacks which became apparent as the popularity of cellular networks grew and the subscriber base expanded. The service providers demanded higher levels of spectrum efficiency and more transparent roaming services. This created the demand for the second generation wireless networks which we study in the next section.

4.4 TWNs: Second Generation

The first generation wireless cellular networks specified the communication interface between the mobile station and the land-station; that is, it specified the air-interface but not the communication interface between the LS and the MTSO. This had far-reaching implications on the system architecture in that the LS and the MTSO had to come from the same vendor, since the communication protocol between the LS and the MTSO was proprietary. The lack of coordination between various vendor switches meant that even though subscribers could make and receive calls within the areas served by their service provider, roaming services between service providers were spotty and inconsistent.

Even though the wireless industry in the United States developed Interim Standard 41 (IS41) to address the roaming problem in first generation networks by standardizing the communication protocol between the MTSOs, the problem still existed in Europe where there were as many as five mutually incompatible air-interface standards in different countries in Europe. This, at a time when Europe was moving towards a model of European economic integration, led the Conference of European Postal and Telecommunication (CEPT) to undertake the development of a continental (read pan-European) standard for mobile communication. This led to the Global Systems for Mobile-Communications (GSM) specification with one of the primary underlying goals being seamless roaming between different service providers.

The term *second generation cellular networks* is a generic term referring to a range of digital cellular technologies. Unlike the first generation networks, all second generation networks have a digital air interface. With an estimated 1 billion subscribers all over the world, the most dominant second generation technology is GSM.

GSM has several salient features. It combines Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA) to specify a hybrid digital air interface. Therefore, unlike AMPS where a logical channel could be specified by specifying just the carrier frequency, a logical channel in a GSM needs to be specified using a carrier frequency (FDMA) and a timeslot (TDMA).

Another important feature of GSM is that it specifies not only the air interface but many other interfaces in the GSM network as shown in Figure 4.5.

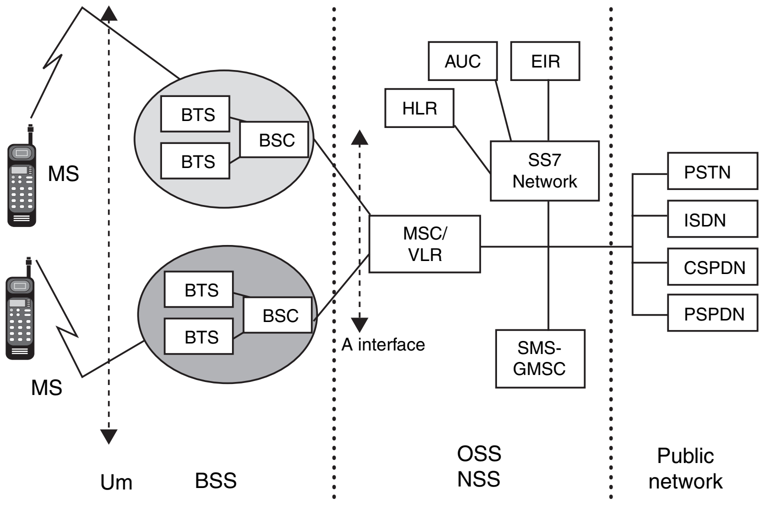


Figure 4.5: GSM System Architecture

The end-user equipment (typically a cell phone) is known as the Mobile Equipment (ME) or the Mobile Station (MS). The term MS refers together to the physical device, the radio transceiver, the digital signal processors, and the Subscriber Identity Module (SIM). The SIM is one of the great ideas to come out of the GSM standard. It is a small electronic card that contains user-specific information like the subscriber identity number, the networks that the user is authorized to use, the user encryption keys and so on. The concept of separating the subscriber specific information from the physical equipment (the phone) allows the user to use their service from a variety of equipment, if they so wishes.

The mobile equipment communicates with the Base Transceiver Station (BTS) which consists of a radio transmitter and a radio receiver and is the radio termination interface for all calls. The interface between the MS and BTS is known as the U_m interface.

The BTS is the hardware which defines the cell (in that each cell has exactly one BTS). It consists of a radio antenna, a radio transceiver and a link to the Base Station Controller (BSC) but it has no intelligence. The intelligence (software) which controls the radio interface sits in the BSC and is responsible for things like channel and frequency allocation, tracking radio measurements, handovers, paging, and so on. Each BSC usually controls multiple BTSs and the interface between these two components

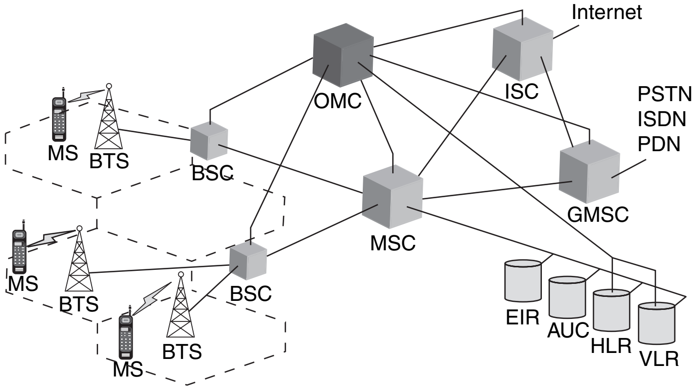


Figure 4.6a: GSM Architecture—*Overview*

is known as the A_{bis} interface. The BSC and the BTSs together constitute the Base Station Subsystem (BSS) of the GSM network. Beyond the BSS exists the GSM core network.

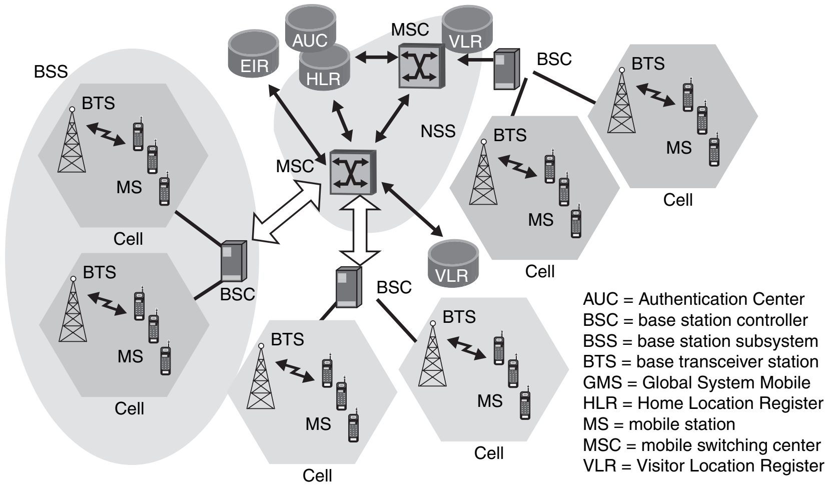

BSS
EIR
AUC
HLR
MSC
VLR
BSC
BTS
NSS
MSC
BTS
MS
Cell
BTS
MS
Cell
BSC
BSC
VLR
BTS
MS
Cell
BTS
MS
Cell
BTS
MS
Cell
BTS
MS
Cell
AUC = Authentication Center
BSC = base station controller
BSS = base station subsystem
BTS = base transceiver station
GMS = Global System Mobile
HLR = Home Location Register
MS = mobile station
MSC = mobile switching center
VLR = Visitor Location Register

Figure 4.6b: GSM Architecture—*Details*

The BSS interface to the core network is the BSC's interface to the Mobile Switching Center (MSC) and is known as the A interface. The MSC is the mobile-aware switch responsible for call routing and call setup. Each MSC is connected to other MSCs and also connects to the public switched telephone network (PSTN). A MSC which connects to the PSTN is known as the Gateway Mobile Switching Center (GMSC). The interface between MSCs is known as the E interface.

The GSM network components described so far are probably sufficient to provide the most basic wireless voice service. However, there are four important databases in the GSM core network which allow the GSM network to provide seamless service to the end user. The first is the Home Location Register (HLR) which stores information on each subscriber that "belongs" to it. This includes information like the subscriber's address, billing information, service contract details, and so forth. The HLR is therefore the central repository of all information regarding the user.

The Visitor Location Register (VLR) is a database in the GSM network that is required to achieve seamless roaming in all service areas in the network. Unlike the HLR which is usually unique at the service provider level, the VLR is one per MSC and keeps track of all users which are currently in the area being served by this MSC. To understand the need for a VLR, consider what happens when a call from the PSTN needs to be terminated on a mobile phone. The PSTN will route the call to the GMSC of the service provider to which the terminating phone-number belongs. The GMSC then queries the HLR regarding this user. The HLR contains a pointer to (the address of) the VLR where the subscriber is currently located. The GMSC can therefore route the call to the corresponding MSC, which would then terminate the call on to the mobile equipment.

The magic of how the HLR knows the current VLR is a complex procedure of location updates as explained in Figure 4.6c. Whenever a mobile equipment detects that

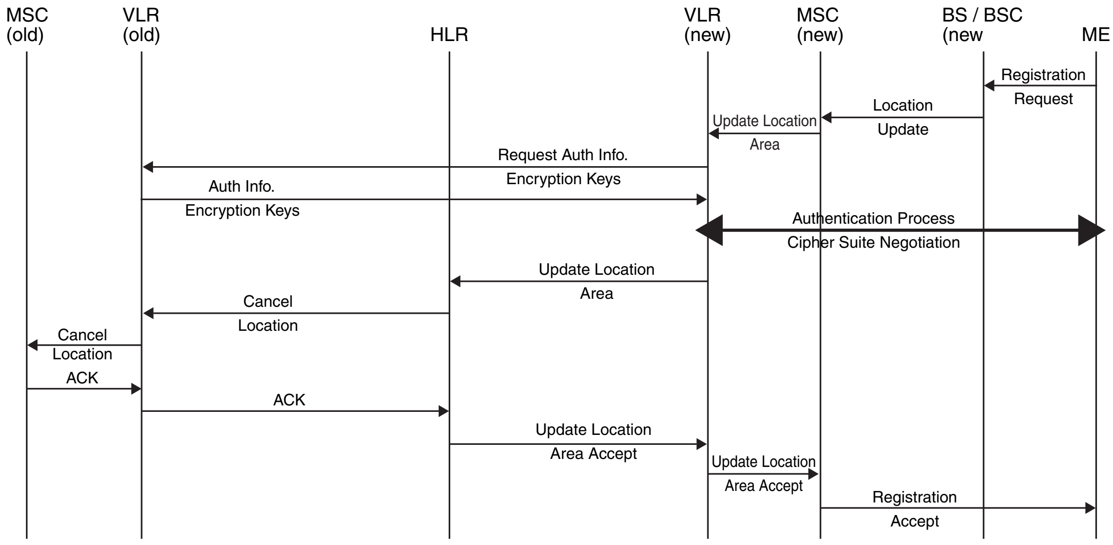


Figure 4.6c: Handoffs and GSM Security

the signal from its current BTS is too low (below a certain threshold), it starts the roaming procedure to connect to the BTS with the strongest signal strength. To do this, the mobile equipment sends a registration request to the new BTS. In turn, BTS sends a location update to its MSC. The MSC then updates its VLR to update information regarding this user. This VLR now contacts the old VLR where the ME was previously registered to get the authentication and encryption keys for this user. Also, the VLR contacts the ME's HLR to update the information regarding this ME. It is the HLR which in turn updates the old VLR to remove the subscriber's identity.

There are two other important databases in the GSM network: the Authentication Center (AuC) and the Equipment Identity Register (EIR) that are needed to provide secure service to the end-user. The AuC and the EIR, like the HLR, are usually unique at the service provider level. The AuC holds the authentication and encryption keys for all subscribers in the HLR and all VLRs in this service provider's network. The EIR on the other hand, keeps track of the user-equipment being used by each subscriber in order to reduce the risk of a SIM card being stolen and used illegally.

The detail and complexity of the GSM standard can be estimated by the fact that the total length of the standard is more than 5000 pages long. The fact that the interface between each network component in GSM is specified allows service providers to purchase different network components from different vendors. Note however that the only interface GSM specifies at the physical layer is the air interface between the MS and the BTS. All other interfaces are specified from Layer 2 above leaving the physical layer implementation to the service provider; for example, the service provider may decide to have the physical interface between the BTS and the BSC as a microwave link or as a fiber optic link depending on the requirements.

4.4.1 Addresses in GSM

One of the unique features of GSM is that it distinguishes between the identity of the subscriber and their phone number.[5] The telephone number of an MS in the GSM network is the mobile subscriber Integrated Services Digital Network (ISDN) number (MSISDN). The identity of the subscriber is established at the time when the subscriber registers for service with a mobile network operator. At this time, the subscriber is assigned an International Mobile Subscriber Identity (IMSI). Both the MSISDN and the IMSI are stored in the SIM[6] and the association of the IMSI and MSISDN is stored in the HLR and is kept secret.

[5] This allows a subscriber to have multiple "phone numbers"—possibly from multiple service providers—and choose among them dynamically.

[6] Note that the SIM is a physically separate and removable component from the ME.

The International Mobile-Station Equipment Identity (IMEI) uniquely identifies the mobile equipment (the phone) internationally. It is allocated by the equipment manufacturer[7] and is registered by the network operator who stores it in the EIR.

To protect the identity of the subscriber, GSM uses a Temporary Mobile Subscriber Identity (TMSI). The TMSI is assigned by the VLR to a subscriber and is used in place of IMSI for the identification and addressing of the MS. Using the TMSI restricts the transmission of IMSI over the air-interface to an absolute minimum, thus protecting the identity of the subscriber. The TMSI is stored on the SIM card on the user-side and at the VLR on the network side. The TMSI is not known to the HLR.

Along with the TMSI, the VLR may also assign another address to the MS: this is the Local Mobile Subscriber Identity (LMSI). The LMSI is assigned to the MS when the MS registers with the VLR and, unlike the TMSI, is known to the HLR. The purpose of LMSI is to act as a short searching key so that messages concerning a particular MS may be tracked faster, leading to short call setup times.

There is another temporary address that is assigned by the VLR to the MS. This is the mobile station roaming number (MSRN). The MSRN is a temporary location-dependent ISDN (phone) number to simplify call routing to the MS.

The cell identifier (CI) uniquely identifies a cell in the GSM network. Note that the CI is different from the base transceiver station identity code (BSIC) which uniquely identifies a BTS.[8]

4.4.2 Call Setup in GSM

A call originating from the PSTN destined to a GSM subscriber proceeds as follows:

1. Call-setup messages reach the GMSC through the PSTN.
2. The GMSC contains a table linking MSISDNs to their corresponding HLR. It uses this table to interrogate the called subscriber's HLR for the MSRN of the called subscriber.
3. The HLR typically stores only the SS7 address of the subscriber's current VLR, and does not have the MSRN. The HLR therefore queries the subscriber's current VLR.
4. This VLR will temporarily allocate an MSRN from its pool for this call and inform the querying HLR of the MSRN.
5. The HLR forwards this MSRN to the GMSC.

[7] Think MAC addresses for Network Interface Cards.
[8] This may happen if the BTS uses directional antennae to create multiple cells.

6. The GMSC uses this MSRN to route the call to the appropriate MSC.
7. When the appropriate MSC receives the call request, it looks up the IMSI corresponding to the MSRN in the call request and then broadcasts a page in the current Location Area of the subscriber.
8. The appropriate ME responds to the paging request.

A call originating from the GSM subscriber destined to the PSTN proceeds as follows:

1. When the user presses the "send" button on their phone, the MS sends the dialed number to the BTS.
2. The BTS relays the dialed number to the MSC.
3. The MSC first checks to see if this number belongs to one of its own subscribers which may be reached "locally" without accessing the PSTN.[9] The MSC can find this out by referring to its HLR.
4. If the called party is a subscriber, the MSC can also determine its current location using the HLR and then forward the call to the appropriate MSC/VLR.
5. If however, the called party is not a subscriber, the MSC uses the PSTN to route the call.
6. Once the MSC receives an acknowledgement from the remote CO, the MSC tells the BTS to allocate voice channels to the MS for this call.
7. The BTS allocates voice channels for the MS and informs the MS about these.
8. The MS can now start the voice conversation using the voice channels allocated to it.

4.5 TWNs: Third Generation

As of the writing of this book, 2G networks like GSM continue to be the most widely deployed wireless networks. It is estimated that GSM alone has over 1 billion subscribers in this world. There is no question that the 2G networks have been hugely successful in satisfying the needs of customers today. So, what is the need for next-generation wireless (3G) networks? That, in fact, is the billion dollar question. Most wireless service providers in the United States and Europe spent billions of dollars in buying[10] the radio spectrum in which 3G was to operate. The expectation at that time was that the bandwidth capacity and the features provided by 3G would soon be needed for next-generation wireless applications.

[9] The motivation is obviously to save cost since using the PSTN means paying someone else money for carrying this call.

[10] Spectrums are owned by governments and are usually auctioned off on an as-demand basis.

As it turns, there have been no new killer applications which would justify moving from 2G to 3G. This, combined with the success of add-on technologies (General Packet Radio Service (GPRS) and so on) have allowed 2.5G (2G + add-on services) to serve the demands of subscribers to date. This has left the service providers who bought the 3G spectrum in a tight spot. Nobody knows for sure what's next for the wireless industry after 2.5G. There are basically two camps—one camp believes that 3G will happen soon enough if only because of the huge amounts of money invested in the technology. This camp is supported by the fact that early deployments of 3G are beginning to appear in Japan. The other camp believes that there will soon be an IP-based 4G wireless technology ready to give 3G a huge low. This camp is supported by the fact that a lot of service providers are holding off 3G deployment waiting for subscriber demand or 4G. The thing about 4G however is that nobody knows for sure what shape and form it will take. So whether 3G will happen or not is still an open question.

In any case, we will take a look at 3G in this section. Figure 4.7 shows the 3G network architecture. The architecture looks very similar to the 2G network but there are

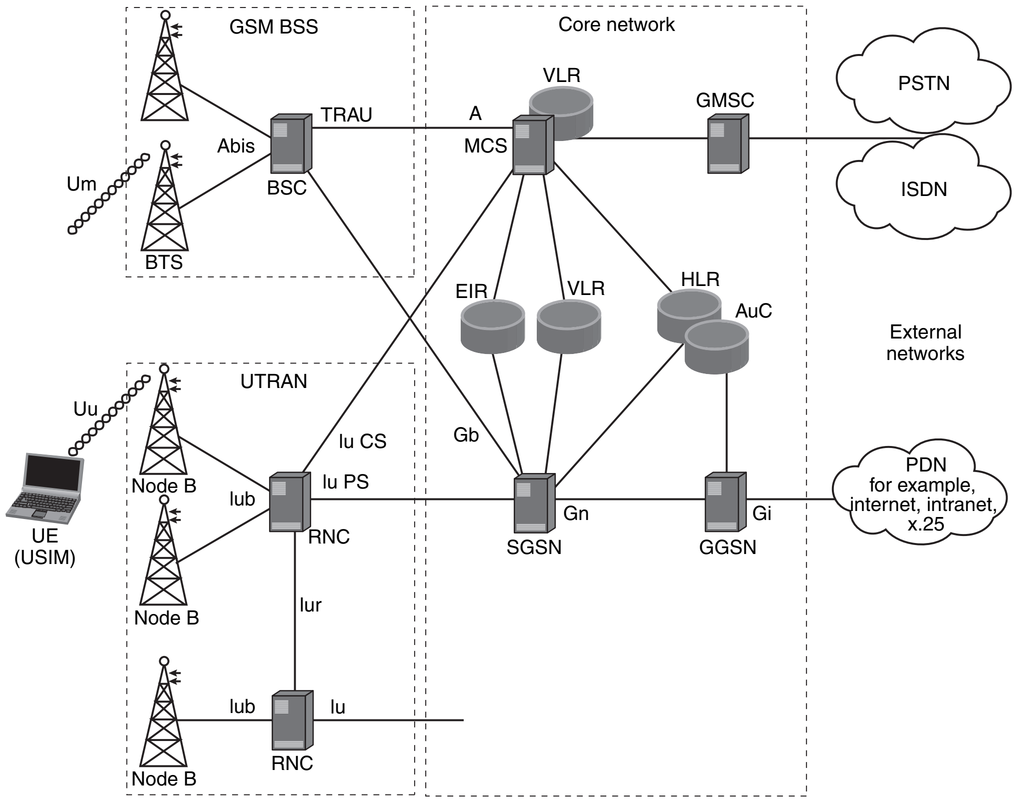


Figure 4.7: UMTS Architecture

important differences. First, the radio interface uses Code Division Multiple Access (CDMA). Even though this is not something new in 3G,[11] it shows the acceptance of CDMA as a superior technology as compared to TDMA in GSM. Second, the network emphasizes the movement of intelligence in the network from the core of the network to the periphery. Third, the architecture shows the integration of the voice network and the IP network.

The Universal Mobile Telecommunications System (UMTS) network architecture is pretty similar to the 2G network architecture with subtle differences. The 3G equivalent of the BTS is known as Node-B. It differs from the BTS primarily in that the air-interface used by 3G is always CDMA unlike 2G where there were multiple standards that could be used for the air interface. The 3G equivalent of the BSC is the Radio Network Controller (RNC). It differs from the BSC in that it takes on many more responsibilities. Most importantly, the role of mobility management is moved from the core of the network to the RNC (effectively the access network; that is, the edge of the network). The RNC connects to the same core network that is used by 2G networks. Therefore, the 3G standard is different from the 2G standard only on the access side: the core network is kept the same.

4.5.1 Connection Setup in UMTS

Since the core network is kept the same as that of the 2G networks, call routing is almost the same as in 2G networks. The differences in the call setup procedure are minimal and are mostly related to who does what since more responsibilities have moved to the edge of the network. We do not therefore repeat the connection setup procedure since for the purposes of this text this is the same as connection setup in 2G networks.

[11] There exist many 2G wireless networks which use CDMA on the radio interface.

4.6 The Overall Picture

We have looked at different generations of voice-oriented wireless networks in this chapter. The first generation wireless networks are now obsolete. However, the second generation and third generation wireless networks continue to co-exist today. To end this chapter, we present a bird's eye view of voice-oriented wireless networks as they exist today:

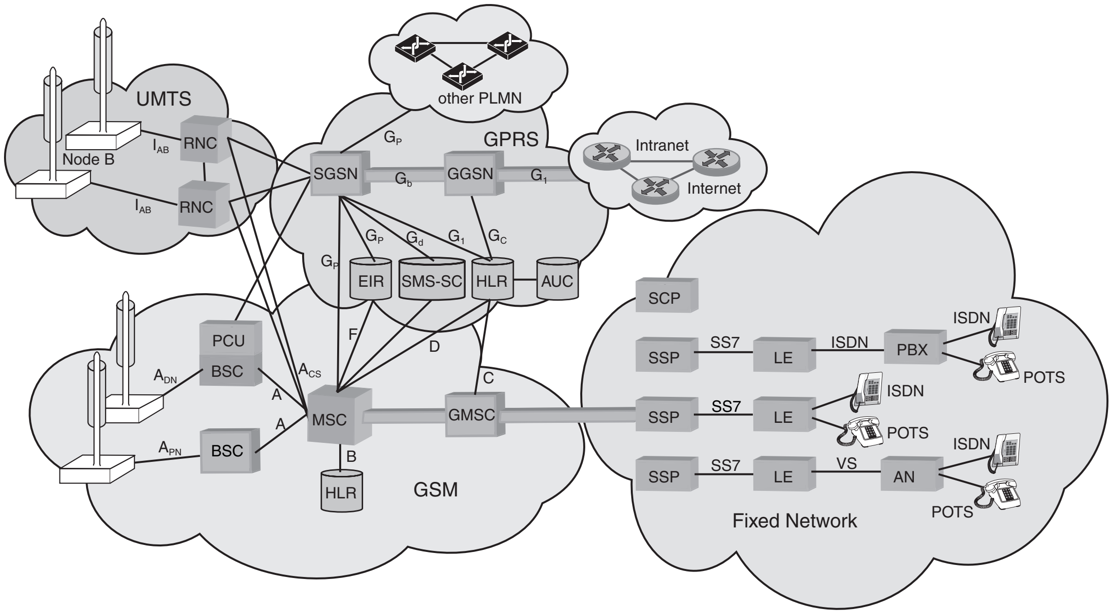


Figure 4.8: Today's Cellular Networks

CHAPTER 5

Data-Oriented Wireless Networks

5.1 WLANs

Advanced Mobile Phone System (AMPS), Global System for Mobile Communications (GSM) and Universal Mobile Telecommunications System (UMTS) were all technologies geared primarily to transfer voice over a Wide Area Network (WAN). With the phenomenal growth in data traffic (think Internet), there has been a demand for wireless networks capable of transferring data traffic along with voice traffic. Just as in the wired world, the field of wireless is seeing the integration of voice and data networks. Second generation (2G) wireless networks, currently the most widely deployed and used, have been enhanced to support data. Such networks are sometimes referred to as 2.5G in order to distinguish them from voice-only 2G wireless networks. Moreover third generation (3G) networks, the next generation wireless networks, have been designed with inherent support to carry both voice and data.

Given the capabilities of 2.5G and 3G to carry data, it may not be apparent at first why there was a need to design another wireless standard. As we would see, 802.11 and 3G are more different than they are similar. Yes, both of them are wireless network standards and yes both of them support both voice and data, but 802.11 is a LAN standard meant to connect wireless clients in a small geographical area whereas 3G aims to provide wide-area (universal) wireless connectivity.

The first widely deployed wireless data network standard has been IEEE's 802.11 standard. The 802.11 standard is a suite of protocols defining an Ethernet-like communication channel using radios instead of wires. Such networks are referred to as Wireless Local Area Networks (WLANs) and the technology is more popularly referred to as Wi-Fi. WLANs allow users to connect to a network (and by extension, to the Internet) without the wires. Put simply, 802.11 is Ethernet (802.3) without the wires. Just as we use 802.3 to form wired local area networks (LANs), we can use 802.11 to create WLANs. On the positive side, since there are no wires to lay down to create the network, setting up WLANs is much easier than setting up LANs. On the other hand, due to the nature of the wireless medium, the packet loss experienced in WLANs is much more than that in wired LANs.

Another distinguishing feature of the 802.11 standard is that it operates in the unlicensed frequency spectrum. This means that 802.11 service providers (popularly referred to as Wi-Fi service providers) do not have to pay a "spectrum usage" fee to their governments. Contrast this with 3G where service providers have spent billions of dollars in purchasing the 3G spectrum which was auctioned by governments worldwide just a few years ago. Operating in the unlicensed frequency spectrum has the advantage of keeping operating costs low but also means no protection from interference caused by other users. This makes the wireless operating environment even more difficult to operate in.

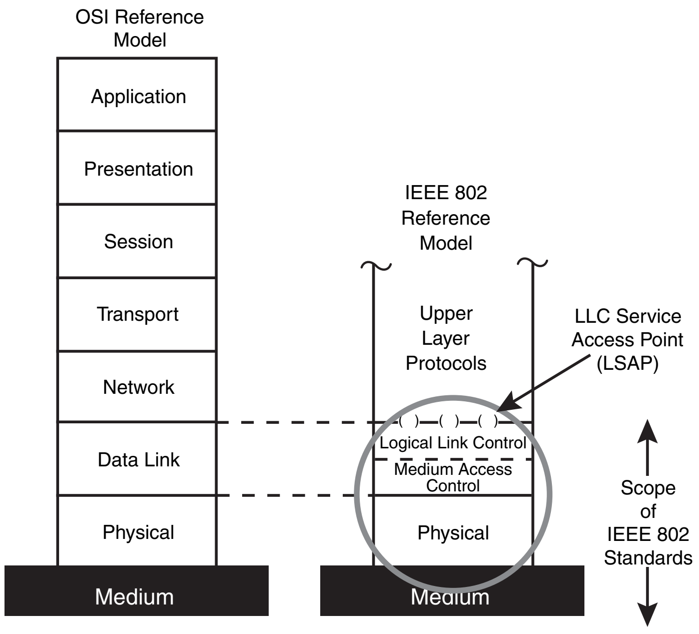


Figure 5.1: 802.11 in the OSI Stack

The 802.11 standard specifies protocols for the physical (PHY) and the media access control (MAC) layers of the open systems interconnection (OSI) stack. Multiple variations of the 802.11 standard define different PHY layers. The first release of 802.11 was made in 1997. It specified a unified MAC layer and three separate PHY layers (Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS) and infrared) that provided for data rates of 1 to 2 Mbps. Since then, the standards have been enhanced to support higher data rates; for example, 802.11b uses DSSS in the 2.4 GHz spectrum to reach data rates up to 11 Mbps whereas 802.11a uses FHSS in the 5.2 GHz range to reach data rates up to 52 Mbps. Figure 5.2 clarifies the 802.11 "alphabet soup."

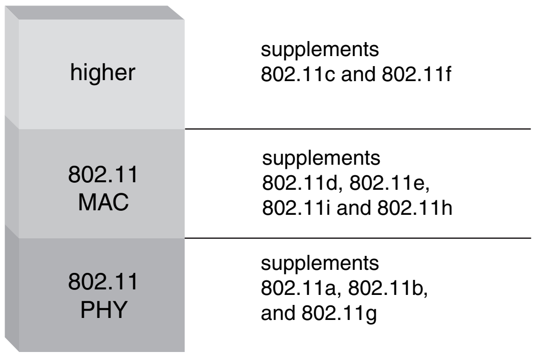

higher
supplements 802.11c and 802.11f
802.11 MAC
supplements 802.11d, 802.11e, 802.11i and 802.11h
802.11 PHY
supplements 802.11a, 802.11b, and 802.11g

Figure 5.2: The 802.11 Alphabet Soup

However, the MAC layer used by all variations of 802.11 is always the same. It is this MAC layer which forms the heart of the 802.11 standard.

A typical 802.11 network consists of four major physical components. First, we have the station (STA). A STA is an end-point of the connection with a wireless interface used to access the 802.11 network. Typical examples of stations are laptops, palmtops and other hand held computers. Figure 5.3 shows laptops as stations.

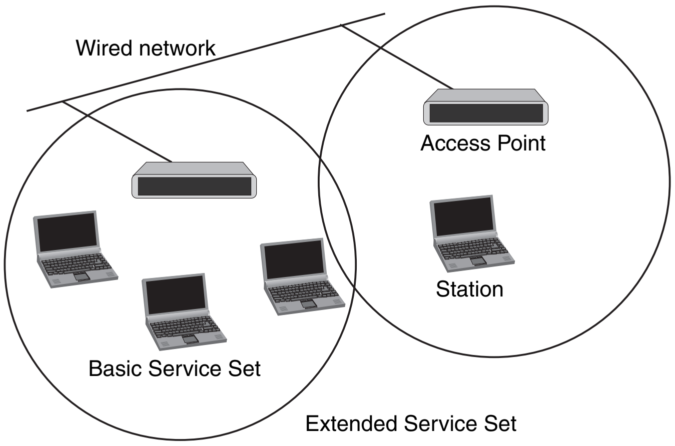


Figure 5.3: 802.11 System Components

Second, we have the Access Point (AP). An access point is basically a Layer 2 bridge which has one wireless interface and one wired interface. It is therefore the AP which connects the wireless LAN (or rather the stations in the WLAN) to the wired LAN. As we saw in Section 4.2, radio propagation effects limit the range of wireless transmissions. In effect, this means that the geographical range served by the base station[1] is limited. This range can be increased by increasing the transmission power level at the base station. However, 802.11 has an additional constraint: it operates in an unlicensed band in the spectrum. By law, the transmission power level in the unlicensed band is restricted. This restricts the range of an access point in 802.11 networks to about 100–300 feet. This area is called the Basic Service Area (BSA). Figure 5.3 shows the BSA of an AP circled. While a STA is within the range of an AP, it has access to the wired network and other stations in this BSA.[2] The set of stations within a BSA which can communicate with each other are called the Basic Service Set (BSS).

Third, we have the wireless medium which actually carries the data between the STAs and the AP. The use of radio waves to carry data significantly complicates the design of the physical layer since the wireless medium presents a much bigger set of challenges than any other medium. To deal with these complications, several physical layer solutions have been proposed and incorporated into the 802.11 standard. Different physical layers satisfy different requirements in different environments. Which physical layer is being used by in an 802.11 WLAN can usually be determined by the letter that follows 802.11; for example, 802.11b uses DSSS; 802.11a uses Orthogonal-Frequency-Division-Multiplexing (OFDM) and so on. To deal with multiple physical layers, all of which use the single 802.11 MAC layer, the 802.11 standard splits the physical layer into two components: the PLCP and the PMD. The Physical Medium Dependent (PMD) is responsible for actually transmitting the frames onto the wireless medium. As is obvious from the name, this layer is different for each physical layer (DSSS, OFDM, FHSS and so on). The Physical Layer Convergence Protocol (PLCP) is responsible for providing a uniform interface of the various physical layers (and the PMDs) to the 802.11 MAC layer. The position of the PLCP in the OSI model is hazy. The PLCP sits between Layers 1 and 2 and abstracts the variations of the physical layers so that the 802.11 MAC can function independently of the physical layer in use.

[1] BTS in GSM; Node-B in 3G, and so on.

[2] 802.11 infrastructure networks require even the inter-STA communication in a BSA to go through the AP. 802.11e does allow STAs to bypass the AP for communicating with each other.

Finally, we have the Distribution System (DS). The DS refers to the wired network that the AP connects to on its wired interface. When a packet/frame destined for the wired network arrives over the wireless interface at the AP, the AP forwards it on its wired interface to the DS. The DS is responsible for delivering it to the right node, which may be a STA on the wired network, another AP or a router. Also, if the STA is mobile and if it moves out of the range of the AP and enters into the range of another AP, this station expects its session to be uninterrupted. Obviously this requires that the two APs be able to communicate with each other. The APs communicate with each other using the DS. In other words, the DS connects various APs to form an Extended Service Set (ESS). The existence of a DS (and hence the existence of an ESS) allows for the possibility of transparent handoff when a STA is mobile. The 802.11 standard does not specify any particular technology for the distribution systems. However, most commercial implementations of 802.11 use Ethernet as the distribution system.

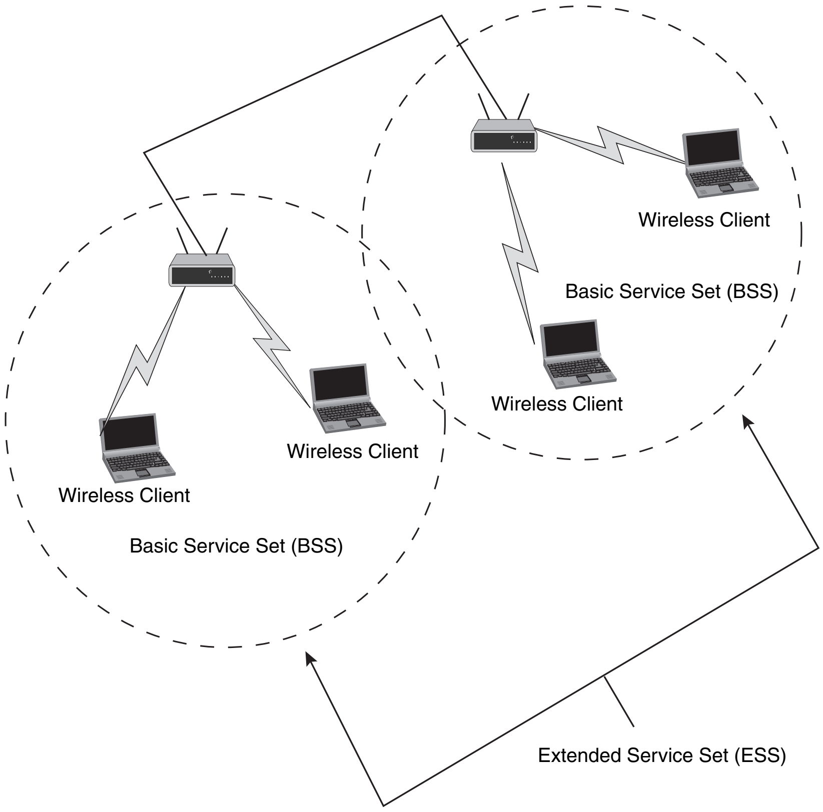


Figure 5.4: 802.11 System Overview

5.1.1: Addresses in 802.11

There are two important addresses in traditional wired data networks—the IP address and the MAC address. IP addresses are used at Layer 3 for routing packets and MAC addresses are used at Layer 2. Each STA in an 802.11 network is uniquely identified by its MAC. This is similar to how end points are identified in traditional wired networks.

5.1.2 Connection Setup in 802.11

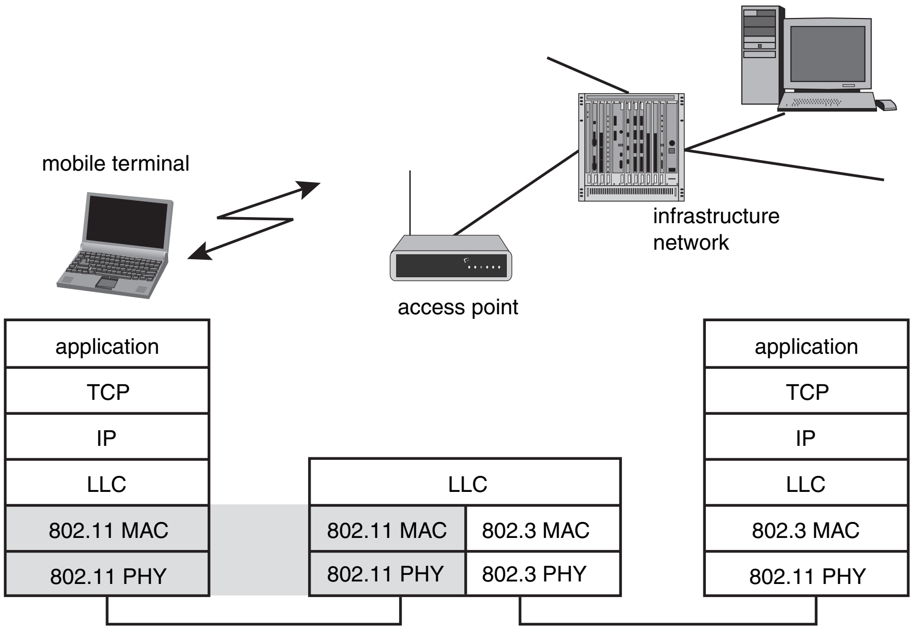


Figure 5.5a: 802.11 System Architecture

As we said at the beginning of this section, 802.11 is a LAN standard which was designed primarily for data communication. The Traditional Wireless Networks (TWNs) were designed primarily for voice and aimed to extend (and inter-work with) the Public Switched Telephone Network (PSTN). The TWN model therefore extends the connection-oriented, circuit-switched model of the PSTN. This means that signaling is used to establish a connection between the endpoints of communication. Establishing a connection involves reserving circuits in the trunks and the switches which would carry the voice call.

The 802.11 standard, on the other hand, fits in more closely with the IP network model than with the PSTN model. The IP networking model is a connectionless,

	TWNs	WLANs	MANETs
Frequency Spectrum	Licensed	Unlicensed	Can use either
Services	Primarily Voice	Primarily Data	Primarily Data
Geographical Coverage	WAN	LAN	PAN
End-User Service	End-to-End	Last Hop (Link-Layer)	End-to-End
Legacy	PSTN-based	IP-based	IP-based
Core Infrastructure	Circuit Switched	Packet Switched	Not Used
Air Interface	Packet Switched	Packet Switched	Packet Switched
Connection	Connection Oriented	Connection Less	Connection Less
Signaling	SS7	N.A.	N.A.
Bandwidth	Reserved using signaling	per-packet contention	per-packet contention
Dominant Standards	GSM, UMTS	802.11, HIPERLAN	In Research phase

Figure 5.5b: Wireless Networks Comparisons

packet-switched network where each packet is routed independently and there is no concept of a connection. The term connection setup therefore becomes irrelevant in an 802.11 network. Since the 802.11 standard has been designed with the IP network model in mind, there is no concept of an end to end connection between the communicating end-points and no resources are reserved in the network. Instead, each STA competes to transmit each and every packet. There is, however, a concept of association between the STA and the AP in 802.11 networks.

We will look at the process of association in a moment, but before that we need to understand what association is and why is it used in WLANs. Remember that TWNs used location management to route incoming calls to the end-user. The exact location of the mobile station (MS) (the exact cell in which the user currently is) is determined only when needed; in other words, only when there is an incoming call for that mobile. To find out the exact cell of the MS during call setup, the mobile switching center (MSC) requests a set of base transceiver stations (BTSs) to broadcast a page request in their respective cell. The MS is then expected to send a page response back to the BTS. This BTS then informs the MSC that it has located the desired MS. This locate-when-needed approach works for TWNs because they are connection-oriented networks where the connection establishment is done once at the beginning of a call and then all voice follows the same path. Contrast this with the datagram (packet-oriented) approach in IP-based networks where each packet is routed independently. 802.11 is designed to work in such an environment. Using a locate-when-needed approach for each packet is obviously too much overhead and therefore not an option. What is needed is for the DS to "know" the location of the end-user at all times so that it can route the packets destined to that STA whenever such packets arrives. This is achieved by the AP. Whenever a STA is within the BSA of an AP, it associates with the AP. Think of the association process like a STA registering its location with the AP. Now, when the DS has a packet meant for a STA, it knows which AP to send the packet to in order to deliver the packet.

This is what we describe in this section. It is however important to reemphasize that 802.11 connections or associations work at the link layer and an 802.11 STA which is associated or connected with an access point may or may not be actively receiving or transmitting data. All this association or connection means is that the STA can transmit and receive data using this AP. A GSM/UMTS connection, on the other hand, is an end to end concept and end points which are connected are in fact actively transmitting or receiving data.

When a STA wishes to connect to an 802.11 network, it broadcasts a “Probe Request” message. This message contains the Service Set Identifier (SSID) of the network (ESS) that this station wishes to connect to.[3] An AP which receives the probe request message may reply back with a “Probe Response” message if it wants to allow this station to connect to its network. In effect, the access point serves as a gatekeeper to the network, deciding which stations to allow access to the network. For this purpose the access point may use several rules. For example, it may allow only those stations to connect which explicitly specified the correct SSID in the probe request message, or it may allow stations with only certain MAC addresses, or it may use the 802.11 authentication process for this purpose. On receiving the probe response message, the STA starts the authentication process. The aim of the authentication process is to establish reliably the identities of the communicating stations. In the case of 802.11, the aim of the connection setup is to establish link layer connectivity and therefore the aim of 802.11 authentications is to ensure the identities of the AP and the STA (the communicating stations involved in establishing the link layer connectivity). Note that this is very different from TWN authentication, which aims to establish end to end authentication because TWNs provides end to end connectivity. On the other hand, since 802.11 is a link layer standard it aims to provide link layer authentication. 802.11 specifies two authentication processes: open system authentication (OSA) and shared key authentication (SKA). We look at these authentication processes in much more detail in Chapter 7.

[3] The SSID field may be left blank in order to indicate that the station wishes to connect to any network that it finds.

The next step in the process is association. The aim of the association process is to establish a logical connection between the STA and the access point. The association process starts with the STA sending an association request to the access point. This request contains parameters like the capability info and the rates that the STA can support. The access point responds with an association response message which may accept or reject the association depending on the parameters provided in the association request message. Once a STA is associated with the access point (and therefore the BSS), the network now knows the location of this STA. This allows the network to deliver data sent to the STA. This is so because only after registering with the AP can the (DS) know that a particular mobile node is being served by one of the APs in an ESS. Since traffic can now flow bi-directionally between the STA and the network, the link layer connection is now established at the completion of the association process.

5.1.3 Media Access

In Section 4.1, we said that wireless is a shared medium. This means that all stations which wish to access the wireless medium should have some rules (a protocol) to decide which station should transmit when multiple stations compete for the medium. The protocols which define these rules are known as media access control (MAC) protocols.

TWNs also use MAC protocols but the role of these protocols becomes much more significant in connectionless networks like IP. In TWNs, MAC protocols[4] are used for resolving contention in the MS to BTS signaling path. The TWN model categorizes the air interface into four broad categories: BTS-to-MS signaling path, BTS-to-MS voice path, MS-to-BTS signaling path and MS-to-BTS voice path. The BTS-to-MS direction is referred to as the downlink and the MS-to-BTS direction is referred to as the uplink. Voice calls in TWNs are explicitly allocated a channel.[5] There is therefore no contention for the wireless medium as far as the voice traffic is concerned. For the signaling traffic in the downlink, there is only one transmitter: the BTS. Again, there is no contention here. MAC protocols are required in TWNs only for signaling in the uplink direction. A typical example of uplink signaling is call initiation from the MS.

In WLANs, MAC protocols take a center seat. Since WLANs are designed to work with a connectionless network in mind, where each packet has to be routed independently, this means that every packet that is transmitted over the air interface has to

[4] Slotted ALOHA in GSM and a random-access + random-back off model for UMTS.

[5] A channel may be a specified by carrier frequency (as in AMPS), a carrier frequency + a timeslot (GSM) or a code (UMTS).

contend for access to the wireless medium. This makes MAC protocols extremely important in WLANs. The 802.11 standard is one such MAC protocol which is based on the 802.3 (wired Ethernet) protocol.

The 802.3 protocol specifies a Carrier Sense Multiple Access with Collision Detection (CSMA-CD) MAC. Collision Detection in 802.3 is simple. It involves measuring the peak voltage on the wire or cable and comparing it with a threshold. If the measured value is greater than the threshold, a collision is assumed to have occurred—collision detection (CD). CD works pretty well for a wired scenario since two packets collide to add up voltages and since there is very little, if any, noise present in the wire (thanks to the shielding). On the other hand, 802.11 works in the wireless medium where the channel is inherently open to multiple noise sources in the environment. It is therefore unreasonable to rely solely on voltage measurement as a means to detect collisions.

As a result, 802.11 employs virtual carrier sensing to detect whether the channel is idle. This mechanism exploits the fact that wireless is inherently a broadcast medium. The Request To Send (RTS)/Clear To Send (CTS) packets which are sent by a node before it starts data transmission contain the Network Allocation Vector (NAV) time. All nodes which can hear these packets therefore know that the channel is busy for the time mentioned. Hence, the stations have detected a busy channel and achieved carrier sensing. Once this time period expires, the node uses the physical carrier sensing to further ensure that the channel is idle.

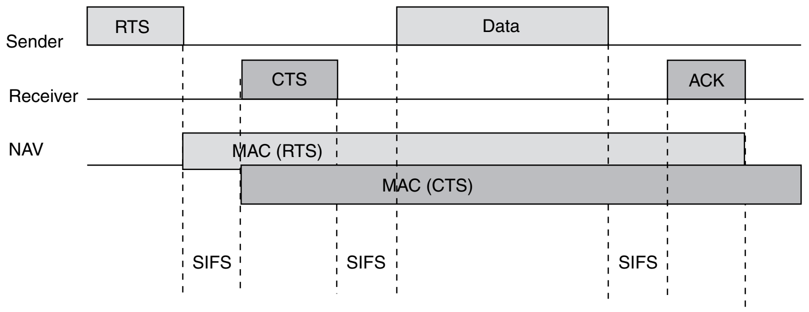


Figure 5.6: 802.11 MAC

802.11 MAC works based on a four-way RTS-CTS-DATA-ACK exchange. It is a distributed random access protocol wherein any node which wants to transmit some data waits a random time and then senses the channel for a Inter-Frame Spacing (DIFS) period. If the channel is idle, it transmits an RTS packet to the destination node. The RTS packet contains the duration of time for which the transmitting nodes wants to capture the channel: NAV. Since wireless is inherently a broadcast medium, all nodes in the vicinity of the transmitting node hear the RTS and are able to read the NAV: the time-duration for which the transmitting node wishes to capture the channel. All these nodes consider the channel busy for the NAV mentioned.

If there is no collision, the RTS reaches the destination node which responds with a CTS after sensing the channel idle for a Short Inter-Frame Spacing (SIFS) period. The CTS also contains a NAV value, and all nodes which hear the CTS accordingly update their timers to assume the channel busy for this duration. Once the node receives a CTS in response to its RTS, it has guaranteed access to the channel for the NAV duration mentioned in the CTS. Once the node has completed transmitting its data, it waits for an ACK from the base station, upon receiving which the transaction is said to be completed. Note that the destination node also senses the channel idle for a SIFS period before sending the ACK. If the transmitting node does not receive an ACK within a specified time period, it assumes that the data has been lost and will retransmit.

If, however, a node on transmitting a RTS does not hear back a CTS, it assumes that the RTS has been lost due to collision. Note that a node may also realize that a RTS has been lost by physical carrier sensing. On realizing that a RTS has been lost, the node doubles its random time wait (binary exponential back-off) and contends for channel access again.

The reason for using a RTS/CTS before transmitting data onto an idle channel is two-fold: to solve the hidden node problem and to reduce the probability and the overhead of collision. Recall the hidden-node problem from the example in Section 4.1.2. If nodes A, B and C used the RTS/CTS protocol, A would transmit an RTS before beginning its transmission. C would not hear this RTS since it is outside the transmission range of A. However, when B responds with a CTS, this CTS would reach C. C can therefore delay its transmission to avoid collision at B. The duration of the delay is obtained from the NAV in the RTS/CTS messages. The second advantage of the RTS/CTS mechanism is that the probability of collision is reduced since RTS and CTS are very short (~20 bytes) as compared to the average size of data packets.

Note that RTS/CTS are not a required feature in 802.11 MAC and a node can directly send data onto a channel provided the data size is small. Since the size of RTS/CTS packets is smaller than the data they precede, the probability that RTS/CTS sent by two nodes will collide is lower. Furthermore, even if a collision occurs, the overhead occurred is proportional to the RTS/CTS packet size which is much less than the data size. A successful RTS/CTS negotiation captures the channel for the following data and ensures that no other nodes transmit during this time by providing the NAV value to other nodes in the vicinity. Hence RTS/CTS are an integral part of the virtual carrier-sensing mechanism used by 802.11.

5.1.4 Spectrum Efficiency in 802.11

Calculating system spectrum efficiency for 802.11 is a little tricky because there are multiple PHYs specified to work with the 802.11 MAC layer specification. Since spectrum efficiency is a factor of the PHY layer, this means that 802.11a, 802.11b and 802.11g would each have a different spectrum efficiency. As an example, consider 802.11b, the most dominant standard today. 802.11b allows for data rates up to 11 Mbps and occupies 83.5 MHz of the spectrum. However, the 11 Mbps[6] is the shared bandwidth available for each channel, of which there are eleven available. Out of these eleven channels, there are only three which are nonoverlapping. Therefore, the effective re-use factor is three. This means that in a given geographical area, an 802.11 network may be setup so that STAs in the network have an available bandwidth of 11 * 3 Mbps. Therefore, in a geographical area, we have up to 33 Mbps available from an allocated 83.5 MHz. The spectral efficiency could therefore be measured to be 0.395 (33/83.5) bpHz. Compare this with 802.11a which occupies 300 MHz and allows for rates up to 54 Mbps in each channel. Since there are twelve nonoverlapping channels available, in a geographical area, we could have up to 648 Mbps available from 300 MHz. The spectral efficiency could therefore be measured to be 2.16 (648/300) bpHz.

Calculating per-connection spectrum efficiency gets really complicated (and controversial) since there is no concept of an end-to-end connection as in GSM/UMTS. Unlike GSM and UMTS, 802.11 does not dedicate resources. Instead each STA competes for bandwidth on as-needed basis. Since the bandwidth is now shared between all competing STAs, it makes it difficult to calculate exactly how much bandwidth each STA is using. There is yet another problem in calculating the spectrum efficiency. The shared bandwidth for which the STAs are competing is a factor of the packet size that is being used by the STAs as shown in Figure 5.7. This means that STAs which use a larger packet size have access to more bandwidth than STAs using a smaller packet size. Given all these factors, per-connection or per-user spectrum efficiency calculation does not really hold for 802.11.

[6] This is the theoretical limit. Practically, this value can only be as high as 7 Mbps.

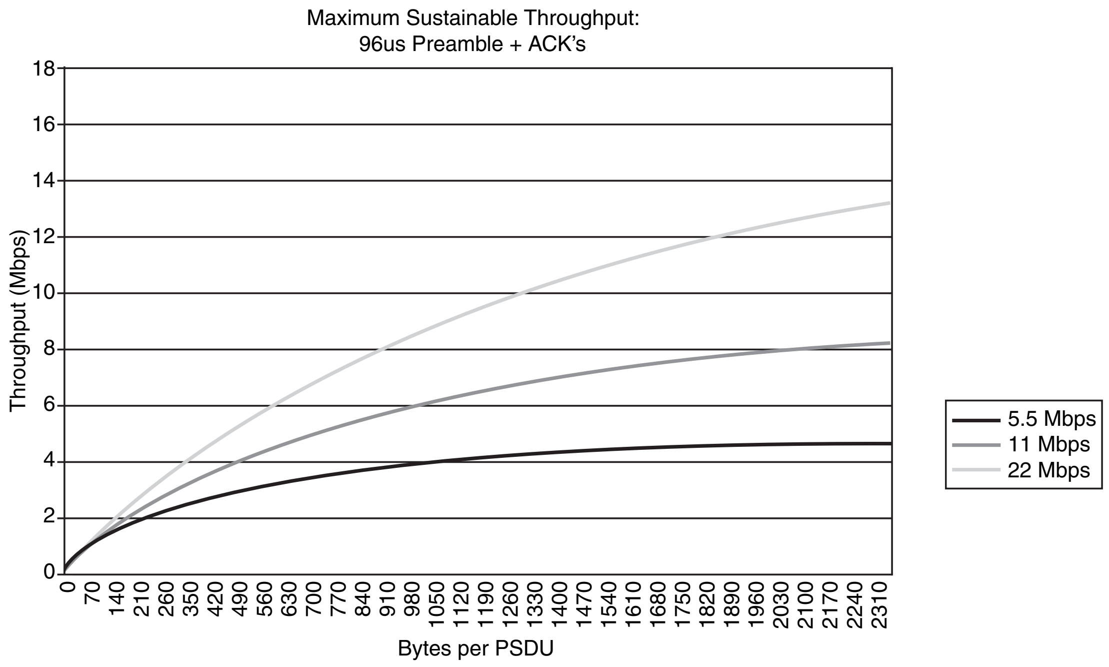


Figure 5.7: Throughput in 802.11 Networks

5.2 MANETs

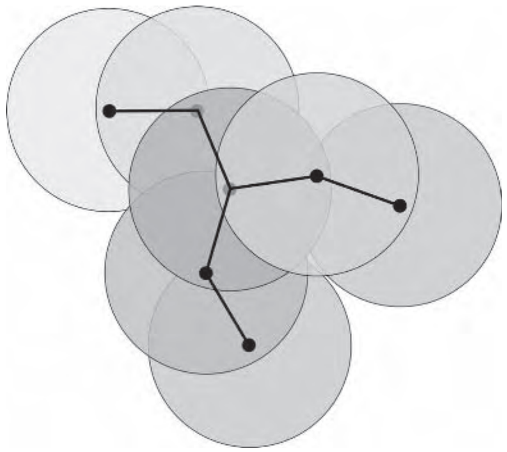


Figure 5.8: Ad Hoc Networks System Overview

Like WLANs, MANETs aim to provide wireless communication in a limited geographical area. Most MANETs cover a much smaller area than WLANs but that is not what distinguishes MANETs from WLANs. The most prominent characteristic of MANETs is that unlike WLANs (and unlike TWNs), they do not rely on any fixed infrastructure to establish communication: instead, wireless nodes co-operate among themselves to establish communication. MANETs are therefore also known as infrastructure-less wireless networks. MANETs are especially attractive for use by the military, emergency service providers and commercial applications where user density is too sparse or too temporary to justify the deployment of any infrastructure.

To put things in perspective, realize that TWNs were characterized by the "intelligence" in the network residing at the core of the network (the PSTN core network) and thus the network was a service (routing, resource-reservation, security and so on) provider for the endpoints. This model changed little in WLANs: even though endpoints became more intelligent and assumed more responsibilities (roaming decisions for example), the core of the network (access points, distribution system and so on) were still responsible for providing most of the services (routing, quality-of-service and so forth).

Ad hoc wireless networks reflect a paradigm shift in wireless communication. Since there is no infrastructure to rely on, all services have to be provided by co-operation between the nodes themselves. To understand what MANETs really are, visualize a classroom of students and a teacher where everyone has a laptop. The aim is that all the students and the teacher in the classroom are connected by a MANET for the duration of the class. Another commonly used example of a MANET is a military network in hostile territory. Since no existing fixed (wired) network is available, MANETs are an attractive option. These examples help us understand what MANETs are but do not really define the MANET concept. For a better understanding of MANETs, we list here some of the salient characteristics of MANETs.

1. Nodes come together to form a network on an as-needed basis for as long or as short a period of time as desired.
2. The network allows for random node mobility. This means that the wireless topology is highly dynamic and may change rapidly and unpredictably as nodes move around and join or leave the network.
3. Communication between the nodes uses the wireless medium. This is an underlying assumption to support mobility of nodes. Operating in the wireless medium leads to the additional constraints:
    a. Error prone environment.
    b. Limited (and variable) bandwidth availability.
    c. Energy—constrained nodes.

4. The network does not use any existing fixed infrastructure. This allows MANETs to be formed whenever, wherever (really ad hoc).
5. A MANET may be an autonomous system enabling communication between its member nodes or it may connect to other networks (wired or wireless) using one of its member nodes as a gateway.
6. If nodes wish to communicate with other nodes outside their radio range, intermediate nodes must act as routers. Such networks are referred to as multihop MANETs. In multihop MANETs, nodes co-operate with each other by relaying each other's packets towards the ultimate destination.

By their very nature, MANETs are challenging to design. We will look at some of the most significant problems facing MANETs today. Keep in mind though that MANETs are an active area of research today and there are hardly any standardized protocols for MANETs. We therefore look at the underlying challenges and concepts of MANETs.

5.2.1 MAC for MANETs

Since there are no standardized MAC protocols for media access control (MAC), we look at how 802.11 supports MANETs. Figure 5.9 shows a MANET formed using the 802.11 MAC protocol. While discussing WLANs, we said that 802.11 supports two modes. We looked at the infrastructure mode in Section 4.5. We look at the Independent Basic Service Set (IBSS) mode here. The IBSS mode which supports MANETs basically removes the AP and the DS from the system architecture, allowing wireless networks to be formed ad hoc. In theory the 802.11 MAC should work perfectly well in this case since the MAC is independent of the AP and DS. In practice, things are a little different.

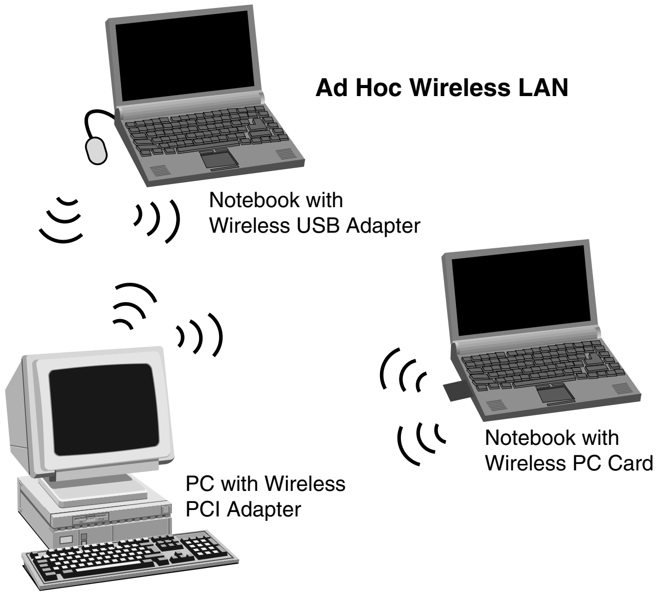

Ad Hoc Wireless LAN
Notebook with Wireless USB Adapter
Notebook with Wireless PC Card
PC with Wireless PCI Adapter

Figure 5.9: 802.11 Ad Hoc Wireless Network

We discussed the 802.11 uses the RTS/CTS mechanism for resolving the hidden-node problem. One of the problems with the

RTS/CTS mechanism is that it does not help in resolving the hidden node problem in a multihop MANET system. To understand the issue we need to understand two important terms. Interfering range (sensing range) refers to the maximum distance at which two wireless nodes can interfere with (or sense) each others' transmission. Communication range refers to the maximum distance at which two wireless nodes can communicate with each other without suffering major losses. The interfering range is larger than (typically double) the communication range.

Figure 5.10 shows a simplified multihop MANET. Assume that node 1 wishes to communicate with node 5. Since they are not within each others' communication range, they need to use the routing services of other nodes (2, 3 and 4) in the network. When 1 wants to send data to 5, it sends the data to 2 and expects it to forward it to 5. Further, assume that each packet is preceded by a RTS/CTS exchange in order to avoid the hidden node problem. At some point during the communication session, 4 would be forwarding packets to 5 and 1 would have to send new packets to 2. So, 1 would send a RTS message to 2 but 2 would not be able to send a CTS back to 1 if 4 is within its interfering range. In other words, since 2 can sense 4's transmission, it has to defer its transmission to 1. This is the exposed node problem which makes the 802.11 MAC unsuitable for multihop MANETs. Another problem with the 802.11 MAC when used in multihop environments is that the RTS/CTS mechanism does not always solve the hidden node problem. The underlying assumption for RTS/CTS to solve the hidden node problem is that all nodes must be within each other's interfering range. While this usually holds true for infrastructure BSS, it rarely holds true for MANETs.

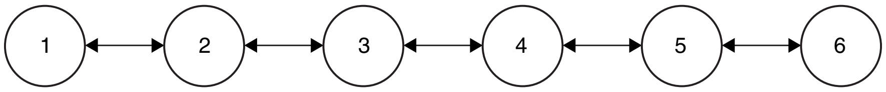


Figure 5.10: Multihop Ad Hoc Networks

5.2.2 Routing in MANETs.

One of the most studied and researched topics in wireless telecommunications today is MANET routing. This in itself is an indication of how tough the problem really is. To appreciate the complexity of the problem, let's take a step back and see what routing is and how it is accomplished in TWNs and WLANs. Recall from our earlier discussion that TWNs use the SS7 signaling network of the PSTN to route calls and WLANs use the IP network consisting of routers to route each packet independently. The common characteristic is the use of a fixed infrastructure for routing even though the end-user is mobile. In MANETs however, because of the very nature of the network, we do not have access to a fixed infrastructure: the end-users (and therefore the routers themselves) are mobile. So, not only do the routes change dynamically but the routers themselves may disappear altogether (when they leave the network). It is this dynamic nature of the network that makes routing extremely complicated in MANETs.

Most MANETs would typically use the IP stack model for establishing a network. The IP world follows a datagram (packet-switched) connectionless routing model. An oft-used analogy is the postal system. Each letter carries its destination (and source) address and is routed to its final destination independently of other letters. When the letter is posted into the postal network, it is first routed to the correct country, then to the correct state, then the city, followed by the locality and finally to the correct street address. Realize that this model works efficiently since none of the involved components is mobile. Consider for example what would happen if the destination address was not fixed geographically. How would the postal network know where to send the letter? This is the problem introduced by user mobility. To make matters worse, consider what would happen if the post offices themselves were not fixed geographically. How would the post office which collects the letter from the user know where to send that packet to? This is the problem when the routers themselves are mobile.

MANET routing protocols are usually classified into two broad categories. Proactive routing protocols aim to maintain routes to all nodes in the network (even if they are not needed). In other words, these protocols practically aim to map the network. This is achieved by the routers (nodes) in the network periodically exchanging routing updates. These updates are then used to maintain up-to-date routing tables in each of the routers (nodes). Whenever a node wishes to transmit a packet, it looks up its routing table and transmits to the appropriate next hop. The advantage of these protocols is that they have very low latencies. The problem is that they require a lot of overhead (in terms of processing power, bandwidth and so on) for maintaining routes which may never be used. Examples of such protocols are Dynamic Destination Sequenced Distance Vector Routing (DSDV) protocol and Optimized Link State Routing (OLSR) protocol.

Reactive routing protocols, on the other hand, work on an on-demand basis where the route to the destination is discovered if and when needed. This reduces the overhead of proactive protocols but increases the latency in packet transmission time, since the route discovery mechanism has to complete before the transmission of the first packet can begin. It is also possible that in the time that it takes to determine the route to the destination, the route becomes invalid (due to a node moving out of the network). Examples of such protocols are Dynamic Source Routing (DSR) protocol and Ad Hoc On-demand Distance Vector (AODV) routing protocol.

There is a third category of routing protocols known as *hybrid* routing protocols which work by combining the proactive and the reactive approaches to optimize performance. Such protocols create a hierarchical network by grouping nodes in close proximity into clusters (a Tier 1 network). Each cluster has a cluster-head which acts as the gateway (default router) for all other nodes. The cluster-heads of different clusters then communicate with each other, thus forming a Tier 2 network. The hybrid approach uses a reactive routing approach at the Tier 1 network (inside the cluster) and a pro-active routing approach at the Tier 2 network (between clusters). This optimizes performance because "minor" node mobility changes routes frequently inside a cluster but the inter-cluster routes are affected only if the nodes travel a large distance. Hybrid protocols are therefore extremely effective in scenarios where nodes move frequently but within a small geographical region.

5.2.3 Address Allocation in MANETs

To carry on with our postal system analogy, we said that if the end user were mobile, it would be tough to route the letter to them. Think about this though: if the end-user were mobile, how would the sender know what their address is in the first place? This is the problem of address allocation in dynamic environments. Before we move on, it is important to point out that there are degrees of mobility. Consider for example that you live in New York and wish to take a vacation for a few months to Hawaii, but you still want to receive your mail while in Hawaii.[7] This can be achieved if you inform your local post office of your temporary address in Hawaii and request them to forward your letters[8] to the temporary address. This is the approach taken by mobile IP. Each user has a home address and a home agent: the local router. When the user moves to a new network it is assigned a temporary address and a foreign agent (a router in this new network) and the user updates its home agent with this information. All packets destined to the user are routed to the home-agent as usual. The home-agent then forwards these packets over the IP network to the foreign agent who then delivers the packets to the end-user.

The underlying assumption of mobile IP is that there exists a home agent. This is not always true for MANETs and this makes the address allocation problem even tougher. Realize that in wired IP networks, IP addresses are usually assigned by using a Dynamic Host Configuration Protocol (DHCP) server. Whenever a new node joins the network, it requests the DHCP server to assign it a valid (and unique) IP address. In

[7] I can't think of any reason why you may want to do this, but it serves as a good analogy anyway.
[8] Your local post office may or may not offer this service, but it is possible in theory.

an ad hoc network however, there is no centralized server. One possible solution is for the cluster head to assign a unique IP address to each node in the cluster. Since the IP address assigned is private to this MANET, this approach works as long as the nodes in the MANET only need to communicate amongst themselves. If the node wishes to communicate to other networks (like the Internet) then the gateway node is responsible for performing Network Address Translation (NAT), which converts private IP addresses to globally unique IP addresses. However, this approach works only if the network uses a hierarchical routing approach and each cluster has appointed a cluster head. In a flat topology MANET where there is no cluster head, each node which joins the network chooses an IP address at random and then performs a Duplicate Address Detection (DAD) process to ensure that the IP address that it has chosen is unique.

5.2.4 Security in MANETs

The challenges we discussed in the preceding sections make MANETs an extremely unsecure environment to operate in. The absence of any trusted authority in the network makes these networks extremely susceptible to security attacks. We look at these problems in Chapter 8.

5.3 Wireless Networks in the Near Future

With 2G being the largest deployed wireless standard today, 3G being projected as the most widely deployed and 802.11 being the most anticipated, what will wireless networks look like in the near future? I am not going to be a fool and make predictions here since history as shown us that technology predictions usually look stupid in retrospect. What I will say is that if anyone tells you that any one of these technologies is going to "win" and wipe out the other two, they probably don't know much about the business side of how technology works. I believe that for quite some time to come, we will have all these technologies (and maybe some more: 4G?) co-existing. 2G networks will continue to exist just because they are so widely deployed. 3G networks are going to be deployed because they offer enhanced data services[9] and hold the promise of wireless connectivity anytime, anywhere on the planet (and beyond?). 802.11 is here to stay too, since it serves a niche market: wireless local area networks. Most people, therefore, agree that the future will see the co-existence of these technologies.

[9] Some may argue that the reason 3G will be deployed is not so much technical as economical: service providers cannot afford to discard a technology in which they have already invested billions of dollars: Think spectrum auctions.

CHAPTER 6

Security in Traditional Wireless Networks

6.1 Security in First Generation TWNs

In Chapter 4, we discussed AMPS as an example of a first generation TWN. These networks were designed with very little security.[1] Since the AMPS radio interface was analog and since AMPS used no encryption, it was relatively simple for a radio hobbyist to intercept cellular telephone conversations with a police scanner. In the AMPS network, for the purposes of authenticating itself to the network, the mobile station sends the Electronic Serial Number (ESN) that it stores to the network. The network verifies that this is a valid ESN and then allows the subscriber access to network services. The problem with the authentication process is that the ESN is sent in clear over the air interface (obviously, since there is no encryption). This means that a radio hobbyist can not only eavesdrop on cellular telephone conversation but can also capture a valid ESN and then use it to commit cellular telephone fraud by cloning another cellular phone and making calls with it. It was the cellular fraud attack along with the concern for subscriber confidentiality that prompted cellular service providers to demand a higher level of security while designing second generation TWNs.

6.2 Security in Second Generation TWNs

One of the prominent design decisions of second generation TWNs was the move from an analog system to use of a digital system. This design decision led to a significant improvement in the security of the system. The use of a speech coding algorithm, Gaussian Minimum Shift Keying (GMSK), digital modulation, slow frequency hopping and TDMA made casual eavesdropping by radio hobbyists significantly more difficult since it required use of much more highly specialized and expensive equipment than a simple police scanner. However, the use of a digital system is only one of the many security provisions that were designed into the second generation TWNs. In this section, we look at security in GSM networks.

[1] To be fair to AMPS designers, they had too many other problems before security became a priority.

Section 4.4 describes the architecture of TWNs in detail. Figure 6.1 shows the high level architecture of GSM, which is the most widely deployed TWN in the world today. Before dwelling on the security of the GSM network, we need to understand the service model of GSM networks. Recall from our discussion of TWNs in Chapter 4 that TWNs evolved from the PSTN with the aim of extending voice communication services to mobile subscribers. Not surprisingly therefore, the GSM network designers aimed to make the GSM "as secure as the PSTN." To appreciate what this means, realize that the PSTN is an extremely controlled environment. The core PSTN network is controlled and regulated by a small group of operators worldwide. For most part the security in the PSTN is ensured by restricting physical access to the network. This is true both at the access network level and at the core network level.

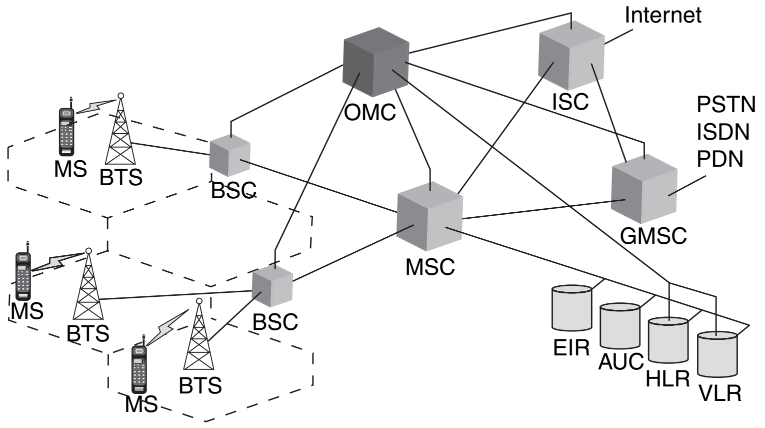


Figure 6.1: GSM Architecture

Since the GSM standard evolved from the PSTN, it carries forward the security philosophy of the PSTN. The network beyond the BTS is considered a controlled environment, since access to this part of the network is controlled by the service provider. It is only the access network (connecting the ME/MS to the BTS) that is considered a hostile operating environment. GSM security therefore aims to secure this part of the network. We look at the details of securing the access network in GSM in the next few sections.

6.2.1 Anonymity in GSM

One of the first things that a ME has to do when it switches on (or roams into) in a coverage area is to identify itself to the network requesting services from the network. Recall from our discussion in Chapter 4 that the IMSI is the unique number, contained in the SIM, by which a subscriber is identified by the network for call signaling purposes. In other words, TWNs use the IMSI to route calls. It is therefore imperative for the network to know where each IMSI is at all times. This functionality wherein the network keeps track of where each IMSI (subscriber) is at any given time is known as *location management*. Even though the details of this topic are beyond the scope of this book, the basic underlying concept of location management is that each time the subscriber crosses a cell boundary,[2] the ME should inform the network about the IMSI's new location. This allows the network to route an incoming call to the correct cell.

In summary, the location update messages from the ME to the network need to carry the identity of the subscriber so that the network knows where to route an incoming call to at any given time. Combine this with the fact that the one-to-one mapping between the IMSI (telephone number) and the subscriber identity is publicly available. This means that if an eavesdropper can capture the IMSI over the air, they can determine the identity of the subscriber and their location. In simpler terms, this means that if you are using a cell phone anywhere in the world, your geographical location can be easily determined. This is not acceptable to most subscribers and therefore this "property" is treated as a security threat in TWNs.

The anonymity feature was designed to protect the subscriber against someone who knows the subscriber's IMSI from using this information to trace the location of the subscriber or to identify calls made to or from the subscriber by eavesdropping on the air interface. GSM protects against subscriber traceability by using temporary mobile subscriber identity (TMSI). Unlike the IMSI which is globally unique, the TMSI has only local significance; that is, the IMSI-TMSI mapping is maintained in the VLR/MSC. When a SIM has authenticated with the network, the network allocates a TMSI to the subscriber. For all communication with the SIM, the network uses this TMSI to refer to the SIM. The use of a TMSI reduces the exposure of IMSI over the air interface to a minimum thus minimizing the probability that an eavesdropper may be able to identify and locate a subscriber.

6.2.2 Key Establishment in GSM

Once the subscriber has identified itself into the network, the next step is to prove to the network that the ME is actually who they are claiming to be: this is the

[2] To be precise, a set of adjoining cells are grouped together to form a location area and the location updates are required only when the subscriber crosses a location area boundary. This reduces the number of messages and thus saves bandwidth.

authentication process. We discuss the authentication process used in GSM networks in the next section. Before we go into that discussion it is important to talk about the key establishment procedure used in GSM networks. As we discussed in Chapter 3, a key establishment procedure is used to establish some sort of a secret or key between two communicating parties. This shared secret then forms the basis for securing the network.

The GSM security model uses a 128-bit preshared secret key (K_i) for securing the ME-to-BTS interface. In other words, there is no key establishment protocol in the GSM security architecture model. Instead each SIM is burnt or embedded with a unique K_i; that is, each subscriber has a unique K_i. Since this is a "shared" secret between the subscriber and the network, it is obvious that the key has to be stored somewhere in the network too. This "somewhere" is the authentication center (AuC) which is basically a database which stores the K_i of all subscribers.[3] It is this shared secret (K_i) between the SIM and the AuC that forms the basis for securing the access interface in GSM networks.

6.2.3 Authentication in GSM

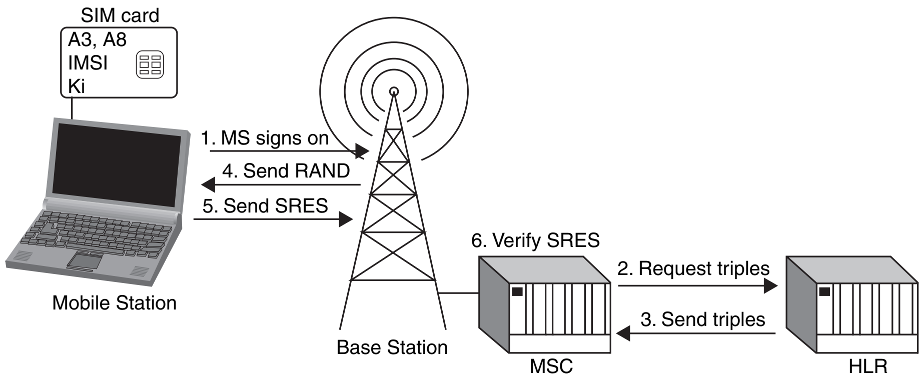


Figure 6.2: GSM Authentication

When a ME first switches on, it searches for a wireless network to connect to by listening to a certain set of frequencies. When it finds a wireless network to connect to, the ME-SIM sends a sign-on message to the BTS requesting access to the network. The BTS then contacts the mobile switching center (MSC) to decide whether or not to allow the ME-SIM access to the network. In order to make this decision, the MSC asks the home location register (HLR) to provide it with five sets of security

[3] In reality, each service provider maintains its own AuC.

triplets. A security triplet consists of three numbers: RAND (a 128-bit random number), SRES (a 32-bit signed response to the RAND generated using the preshared K_i) and a session key K_c (an encryption key generated using K_i). The HLR supplies these triplets to the MSC by using the K_i from the AuC. The MSC then picks one of these five sets of triplets to use for the current "session." The RAND from this triplet is then sent to the ME (via the base station controller (BSC) and the BTS) as a challenge. The ME-SIM is then expected to generate a SRES to this RAND using the A3 algorithm and the K_i stored in its SIM (as shown in Figure 6.3). This SRES is sent back to the MSC (via the BTS and the BSC). The MSC compares the SRES received from the ME to the SRES contained in the triplet it received from the HLR. If the two match, the MSC can safely deduce that the ME has a SIM which contains a valid K_i. The MSC can therefore safely allow the ME access to the network. On the other hand, if the two SRESs do not match, the MSC would not allow the ME access to the network. The GSM authentication process is shown in Figure 6.2.

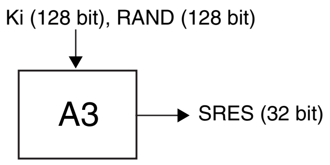


Figure 6.3: GSM SRES Generation

As shown in Figure 6.2, the authentication process is carried out between the SIM and the MSC. The SIM uses the preshared secret K_i that it stores and carries out the A3 and the A8 algorithms to generate the SRES and the session key K_c. It is important to note that the K_i, IMSI and the A3 and A8 algorithms are stored and implemented in the SIM. More importantly, the K_i (which forms the basis of all security in GSM networks) never leaves the SIM.[4]

In the authentication process just described, note the inherent trust relationship between the HLR and the MSC. Such an inherent trust relationship is also present between the BSC and the MSC and again between the BSC and the BTS. This brings us to a very important characteristic of the GSM security model: it aims to secure the wireless part of the GSM network only. In retrospect, this may be considered a flaw, but remember that the GSM network evolved from the PSTN network and the aim of the GSM security designers was to make the GSM network as secure as the

[4] Compare this with first generation TWNs where the ESN was transmitted in clear over the air interface.

PSTN network. Realize that access to the PSTN network was (and still is) very tightly controlled. There are only a very small number of PSTN service providers and therefore getting access to the core network is not trivial. In other words, the core PSTN network is secured by restricting physical access to the network.[5] The GSM network designers carried forward this philosophy. The core network in the GSM architecture refers to the network beyond the BSC and it is considered "secure" since it is controlled by the service provider and access to it is tightly controlled. Therefore, the aim of the GSM security designers was to secure the wireless access network only. However, there is a missing link even if we assume that the core GSM network is secured by the service provider (either by restricting physical access to the network or by other proprietary means). This missing the link is the one between the BTS and the BSC. Remember that this link is not part of the core network. Combine this with the fact that GSM does not specify how the BTS and the BSC need to be connected.[6] In practice, it is common for the BTS and the BSC to be connected by microwave (wireless links). GSM does not specify how to secure this link, thus making it susceptible to attacks.

There is another important characteristic of the GSM authentication process that is worth discussing. In GSM, the authenticating entity is the SIM and not the subscriber per se. In other words, the network authenticates the SIM card and not the subscriber of the SIM card. Remember that the authentication process relies on a preshared secret (K_i) between the SIM and the AuC. During the authentication process, the MSC validates that the SIM trying to access the network has a valid K_i. What happens if a ME is stolen and is used for making calls (and using other GSM services)?

GSM does have some countermeasures to protect against equipment theft. For one, the GSM core network maintains a database of all valid mobile equipment[7] on the network. This database is called the Equipment Identity Register (EIR). If a subscriber loses their ME, it is their responsibility to report it to the service provider. Before authenticating the ME into the network, the MSC also ensures that the ME that is trying to authenticate in to the network has not been compromised. Extrapolating this approach, a service provider may also maintain a list of compromised SIMs. When a SIM is reported stolen, the service provider marks the IMSI and the corresponding K_i[8] as compromised. If a compromised SIM tries to access the network, it is denied access.

[5] The access network of the PSTN on the other hand, is much easier to access as compared to the core network and therefore the PSTN security in the access network is much easier to violate.

[6] GSM just specifies the interface between the BTS and the BSC.

[7] Each ME in the GSM network is uniquely identified by the international mobile equipment identity (IMEI).

[8] There is a one-to-one mapping between the IMSI and the K_i.

Note that when the GSM authentication process completes, it has also established a security context: the session key K_c which can then be used for providing confidentiality in the network. It is the preshared secret key (K_i) between the SIM and the AuC that forms the basis of generating the session key. GSM uses the A8 algorithm to derive a session key K_c from the preshared secret key K_i as shown in Figure 6.4.

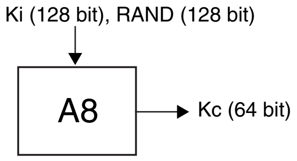


Figure 6.4: GSM Kc Generation

Compare Figure 6.4 with Figure 6.3. The purpose of the A8 algorithm is to derive a 64-bit session key (K_c) given the 128-bit K_i and the 128-bit RAND. On the other hand, the purpose of the A3 algorithm is to derive a 32-bit SRES given the same two inputs (the K_i and the RAND). The important thing to note here is that A3 and A8 are not algorithms per se: they are just labels (reference names) for algorithms. In other words, a service provider is free to use any algorithm that it wishes to generate SRES from K_i and RAND. The GSM specification just uses the name A3 to reference such an algorithm. Similarly, the service provider is also free to use any algorithm that it wishes to generate K_c from K_i and the name A8 is just used by the specification to reference this algorithm. Most GSM implementations combine the A3 and A8 functionality and use a single algorithm to serve both the purposes. The COMP128 algorithm, which is the reference algorithm specified in the GSM specification, takes as input the 128-bit K_i and the 128-bit RAND and generates the 32-bit SRES and a 54-bit number. The 54-bit number is appended with 10-zeros to form the 64-bit session key: K_c. We will see in Section 6.2.3 how this session key is used for providing confidentiality.

GSM allows the service provider to choose an algorithm for A3 and A8 implementation while still ensuring seamless roaming among networks of different service providers. This is an important accomplishment and is achieved because even though the authentication process is carried out between the ME and the servicing MSC, the servicing MSC utilizes the HLR of the ME to authenticate the network. Indirectly therefore, it is the home network of the ME which authenticates the ME into another service provider's network. Since the A3 and A8 algorithms need to execute only at the HLR and the SIM[9] (both of which "belong to" the service provider), they can be proprietary algorithms.

One of the finer details of the authentication process in GSM is the use of five sets of security triplets that the MSC gets from the HLR. Even though only one set of triplets is required for authenticating a subscriber into the network, five sets are requested so as to improve roaming performance. Realize that a ME needs to authenticate with a MSC each time it enters its network from another service provider's network. Instead of contacting the HLR for security triplets each time a ME roams into its coverage area, the MSC gets five sets of triplets: one for the current authentication process and four for future use. This reduces the roaming/handover time and improves system performance.

6.2.4 Confidentiality in GSM

In the previous section, we saw how the GSM authentication process establishes a security context (the session key K_c) when it completes. This session key is used for providing confidentiality over the wireless (ME – BTS) interface. The algorithm used for encrypting packets over the air interface is the A5 algorithm. Unlike A3 and A8 which are just names used by the GSM standard to reference operator-specific algorithms, the A5 is actually an encryption algorithm specified by the GSM standard. The reasoning behind this design decision is the need to support seamless roaming across networks of different service providers. As we saw in Section 6.2.3, the choice of A3 and A8 could be left to the operator since the authentication process is carried out between the SIM and the service providers HLR. The process of encryption on the other hand must necessarily be carried out between the BTS and ME without involving the home network.[10] For achieving seamless roaming between different networks, it is therefore imperative that all service providers use the same encryption algorithm.

The A5 algorithm is basically a stream cipher which generates a unique key stream for every packet by using the 64-bit session key (K_c) and the sequence number of the frame as the input. Since the sequence number of each packet can be easily determined, the confidentiality of the packets depends on keeping the session key (K_c) secret. There is therefore provision in GSM to change the ciphering key K_c: thus making the system more resistant to eavesdropping. The ciphering key may be changed at regular intervals or as required by the service provider.

[9] The A3 and A8 algorithms are implemented in the SIM.

[10] A packet sent from a ME may very well reach its destination without traversing through its home network (packet routing is done by the serving MSC and not the home MSC).

Once the ciphering key has been established between the SIM and the network, the encryption of signaling messages and subscriber data traffic begins as soon as the GSM network sends a ciphering mode request to the ME. Note that unlike the A3 and the A8 algorithms, the encryption algorithm A5 is implemented in the ME.

6.2.5 What's Wrong with GSM Security?

Probably the most glaring vulnerability in the GSM security architecture is that there is no provision for any integrity protection of data or messages. The GSM security architecture talks about authentication and confidentiality but not about integrity protection. The absence of integrity protection mechanisms means that the receiver cannot verify that a certain message was not tampered with. This opens the door for multiple variation of man-in-the-middle attacks in GSM networks.

Another important vulnerability in the GSM security architecture is the limited encryption scope. In simpler terms, GSM concentrates only on securing the ME-BTS interface. We saw in Section 6.2.1 that the reason behind this design decision lies in the evolution of GSM from the PSTN. The fact however remains that the only link which is cryptographically protected in the GSM network is the ME-BTS wireless interface. This exposes the rest of the network to attacks.[11] One of the most exposed links which is not cryptographically protected in the GSM network is the BTS-BSC interface. Since this link is not part of the "core" network and since this link is often a wireless link (microwave-based, satellite-based and so on), it becomes an attractive target for attacks.

The GSM cipher algorithms are not published along with the GSM standards. In fact, access to these algorithms is tightly controlled. This means that the algorithms are not publicly available for peer review by the security community. This has received some criticism since one of the tenets of cryptography is that the security of the system should lie not in the algorithm but rather in the keys. The thinking is that it is therefore best to let the algorithm be publicly reviewed so that the loopholes in the algorithm are discovered and published. Workarounds can then be found to close these loopholes. However, keeping the algorithms secret (like GSM does) denies this opportunity: hence the criticism. To be fair to GSM designers, the GSM specifications came out at a time when the controls on the export and use of cryptography were extremely tight and therefore not making the algorithms public was at least partly a regulatory decision.

[11] Unless the service provider explicitly secures these links.

Even the algorithm used for encryption in the ME-BTS link is no longer secure given the increasing processing power of hardware available today. Using the simplest of all attacks, the brute force attack (which works by trying to break down the security of the system by trying each one of all possible keys), the GSM encryption algorithm A5 can be compromised within a matter of hours. The primary problem is the small key length of the session key K_c. The actual length of K_c is 64 bits. However, the last 10 bits of this key are specified to be 0 thus reducing the effective key size to 54 bits. Even though this key size is big enough to protect against real-time attacks (decrypting packets being transmitted in real-time), the state of the hardware available today makes it possible to record the packets between the MS and the BTS and then decode them at a later time. An important thing to note is that there are multiple A5 algorithms specified in the GSM standard. The first (and probably the strongest) A5 algorithm is the A5/1 algorithm. However, the A5/1 algorithm was too strong for export purposes and therefore the GSM standard specified other A5 variations which are named A5/x, for example the A5/2 algorithm has an effective security of only 2^{16} against brute force attacks (as opposed to A5/1 which has an effective security of 2^{54} against brute force attacks). As we know, brute-force attacks are not the most efficient attacks on the network. It is often possible to reduce the effective security of the system by exploiting loopholes in the security algorithm. For the A5 algorithm, differential cryptanalysis has been shown to reduce the effective security of the system even more. Lastly, the GSM security architecture is inflexible; in other words, it is difficult to replace the existing encryption algorithm (A5) with a more effective algorithm or to increase the length of the key used in the A5 encryption algorithm.[12] In a sense, therefore, the GSM networks are "stuck with" the A5 algorithm.

Another important vulnerability in the GSM security architecture is that it uses one-way authentication where the network verifies the identity of the subscriber (the ME, to be accurate). There is no way for the ME to verify the authenticity of the network. This allows a rogue element to masquerade as a BTS and hijack the ME. Again, to be fair to GSM security designers, at the time of the writing of the GSM standards, it was hard to imagine a false base station attack (an attacker masquerading as the GSM network) since the equipment required to launch such an attack was just too expensive. However, with the phenomenal growth in GSM networks, the cost of this equipment has gone down and the availability has gone up, thus making these attacks much more probable.

[12] It is however possible to increase the effective size of the key from 54 bits to the actual length of 64 bits by removing the requirement of having the leading 10 bits of the key to be all zeros.

A very real attack against the GSM network is known as SIM cloning. The aim of this attack is to recover the K_i from a SIM card. Once the K_i is known, it can be used not only to listen to the calls made from this SIM but also to place calls which actually get billed to this subscriber. The SIM cloning attack is a chosen-plaintext attack which sends a list of chosen plaintexts to the SIM as challenges (RAND). The A8 algorithm generates the SRES to these challenges and responds back. The attacker therefore now has access to a list of chosen-plaintext, ciphertext pairs. If the algorithm used for A8 implementation is the COMP128 reference algorithm and if the RANDs are chosen appropriately, this list of pairs can be analyzed to reveal enough information to recover the K_i using differential cryptanalysis. There are many variations of the SIM cloning attack. In one approach, the attacker has physical access to the SIM card and a personal computer is used to communicate with the SIM through a smart card reader. This approach recovers the K_i in a matter of few hours.

However, it is not always possible to have physical access to the SIM. Therefore, another approach is to launch this attack wirelessly over the air interface. Even though this approach removes the requirement of having the physical access to the SIM (thus making the attack far more attractive), it introduces obstacles of its own. First, the attacker should be capable of masquerading as a rogue BTS. This means that it should be capable of generating a signal strong enough to overpower the signal of the legitimate BTS. Only if this is true would the attacker be able to communicate with the ME. One workaround is to launch this attack when the signal from the legitimate BTS is too weak (in a subway, elevator and so on) The second obstacle arises if the ME is moving. In this case there might not be enough time to collect enough chosen-plaintext, ciphertext pairs to recover the K_i because the inherent latency in the wireless interface increases the time required for each transaction. A workaround to this problem is break up the attack over a period of time. Instead of trying to get all the plaintext, ciphertext pairs in one run, the attacker gets only as many pairs as they can and stores them. They repeat this process over a period of days till they get enough data to recover the K_i.

Yet another variation of this attack attempts to have the AuC generate the SRES of given RANDs instead of using the SIM. This attack exploits the lack of security in the SS7 signaling network. Since the core signaling network is not cryptographically protected and incoming messages are not verified for authenticity, it is possible to use the AuC to generate SRESs for chosen RANDs.

A salient feature of the GSM security architecture is that it is transparent to the subscriber. However this feature sometimes becomes a loophole. There are scenarios where a service provider may choose to use null encryption (A5/0). If a ME is in such a cell, should it be allowed to connect to such a BTS or not? The current design is to allow the ME to connect to such a cell.

6.3 Security in 2.5 Generation TWNs

As we have discussed, the GSM network evolved from the PSTN network and the GSM security architecture followed the same path. GSM security architecture was designed to secure only the last hop (BTS-ME) in the network since the rest of the network was assumed to be a "secure environment" controlled and secured by the service provider. This architecture worked for voice communication and PSTN-based networks because it was relatively easy for the limited number of service providers to maintain a secure environment in the core network.

With the explosive growth in the Internet, 2G service providers upgraded their networks to 2.5G networks to provide data services to their subscribers. These data services basically consisted of connecting the ME to the Internet (that is, various web servers). The 2.5G system architecture looks like the one shown in Figure 6.5.

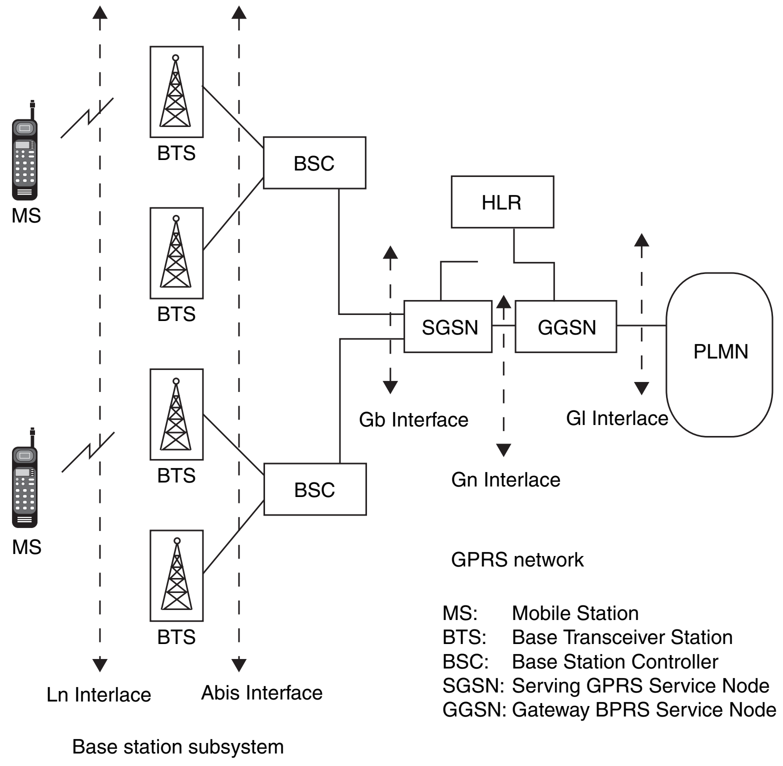


Figure 6.5: GPRS Network Architecture

General Packet Radio Service (GPRS) was basically intended to provide the ME with data-connectivity to various web servers. Since data usually requires more bandwidth than voice, the GSM network achieves this by allocating multiple timeslots to an ME which is trying to access data services.[13] This has an interesting implication on the security architecture of the network. Recall that for voice calls the encryption and decryption happens at the BTS on the network side. For the A5 algorithm, this is possible because the BTS knows the ciphering key K_c and can implicitly deduce the sequence number. These are the only two inputs required to operate the A5 algorithm. In the GPRS architecture, since a ME has multiple timeslots to transmit, it is possible that multiple timeslots are allocated on channels belonging to different BTSs to connect to the network. This may happen for example during roaming as shown in Figure 6.6. This in turn means that the BTS cannot implicitly deduce the sequence number of a packet. To solve this problem, GPRS transfers the responsibility of encryption and decryption on the network side from the BTS to the SGSN. The SGSN is the equivalent of the VLR and MSC. This means that the GPRS architecture effectively prevents (protects against) eavesdropping on the backbone between the BTS and the SGSN too.

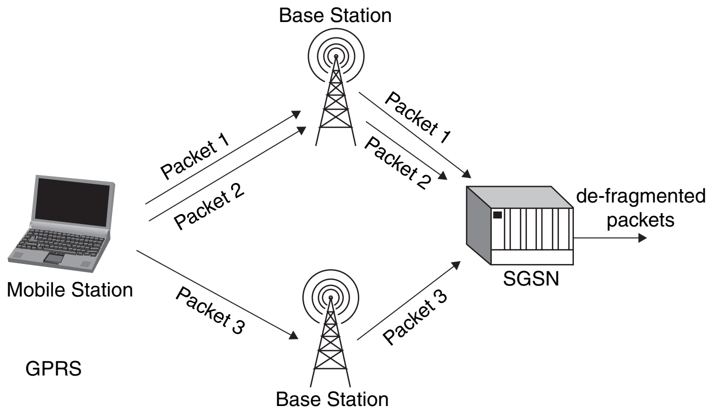


Figure 6.6: GPRS Roaming

[13] Recall that for a voice call, GSM assigns only one timeslot to a ME.

6.3.1 WAP

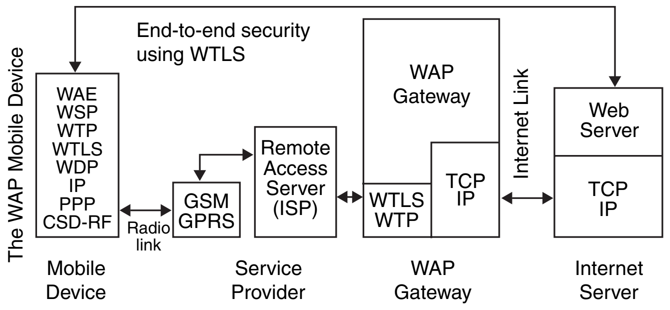


Figure 6.7: WAP—Network Architecture

The GPRS protocol provides a connectivity mechanism for the ME to connect to a data network (Internet). From an OSI layer perspective, GPRS provides Layer 2 (point-to-point) connectivity. What is still required is a set of higher layer protocols (see Figure 6.7). In the wired network, internet applications use the Hyper Text Transfer Protocol (HTTP) and the Hyper Text Markup Language (HTML) to access and retrieve data content (web pages, applets and so on) from web servers. Ideally, the same protocols could have been used over GPRS. This, along with an embedded browser in the ME, would have made the ME a PC-like medium to browse the internet. The problem is that we are operating in a bandwidth-constrained medium and a memory-constrained, CPU-constrained, screen-size constrained end-point (the ME). HTTP and HTML are not optimized for operating under such conditions. This is where Wireless Application Protocol (WAP) comes in.

WAP is an open specification that offers a standard method to access Internet-based content and services from wireless devices such as mobile phones and Personal Digital Assistants (PDAs). The WAP protocol stack is designed for minimizing bandwidth requirements and guaranteeing that a variety of wireless networks can run WAP applications. The information content meant for the ME is formatted suitably for the ME's small screen, and a low bandwidth, high latency environment; i.e., the Wireless Application Environment (WAE).

Figure 6.8 shows the WAP programming model. The client is the embedded browser in the ME and the server may be any regular web server. The new entity in the architecture is the WAP gateway. The embedded browser connects to the WAP gateway and makes requests for information from web servers in the form of a normal Universal Resource Locator (URL). The gateway forwards this request to the appropriate web server and gets the information using HTTP in HTML format. Note that the gateway

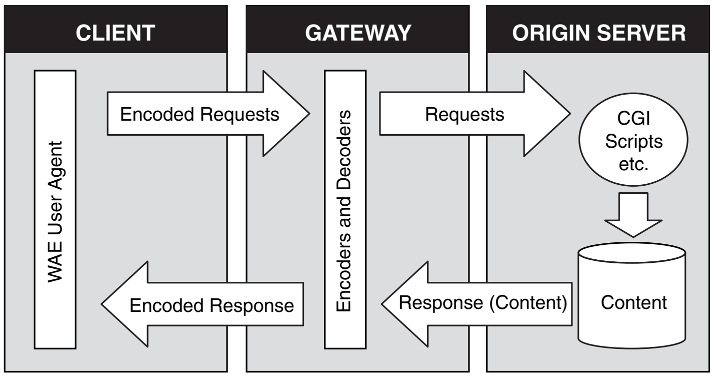


Figure 6.8: WAP—*Overview*

to the web servers is usually a wired link, which is appropriate for using HTTP and HTML. The role of the gateway is to reformat the content from the web server suitable for transmission in a WAE and for display on a ME. The language used for creating this content is called Wireless Markup Language (WML) which is optimized for low bandwidth, high latency connections. To summarize, the WAP gateway is the translator between HTTP, HTML on the web server side and WTP, WML on the ME side.

So, together WAP and GPRS allow the ME to connect to the Internet. Since the Internet is a huge uncontrolled network of haphazardly connected (and growing) nodes, this breaks one of the biggest assumptions of the GSM security architecture—that the core network is a controlled secure environment. In this new operating environment, securing just the last link is not enough. Instead, an end-to-end security architecture is desired. This end-to-end security is achieved by the Wireless Transport Layer Security (WTLS) layer in the WAP stack.

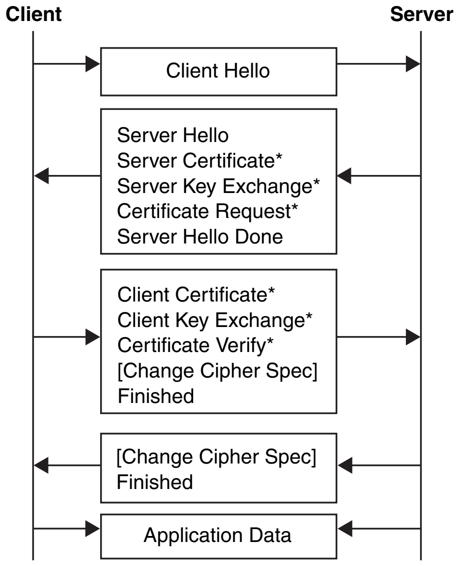


Figure 6.9: TLS in WAP

WTLS is modeled along the lines of Secure Sockets Layer (SSL)/Transport Layer Security (TLS). The reason for designing a new protocol along the lines of TLS and not using TLS itself is optimization. First, TLS was designed to be used over a reliable transport layer (such as TCP) whereas WTLS needs to operate over an unreliable datagram transport where datagrams may be lost, duplicated or re-ordered. Second, the WTLS protocol was modified to cope with long roundtrip times and limited bandwidth availability typical of the wireless environment. Finally, WTLS has been optimized to operate with limited processing power and limited memory of the ME. Figure 6.9 shows a WTLS session. Because of the strong semblance of WTLS to the TLS protocol, the reader is referred to Section 3.5 for more explanation.

6.3.2 Code Security

For most part, network security is concerned with transactional security on the network. In other words, we talk about securing the link(s) in the network using encryption, securing access to the network using authentication and so on. The merging of the GSM network with the Internet, however, adds another dimension to the concept of network security. The ME in a GPRS network can "browse" the Internet. In the Internet architecture, web servers sometimes download applets (short programs) on to the client (ME) over the network. These applets then execute at the client (ME). Consider what happens if the applet is a malicious piece of code or it simply has a bug which can harm the ME. To protect against such attacks, it is extremely important that the applets (or other programs which are downloaded from remote sites but execute on the client) be secure. For more on this topic, see Section 1.7.3.

From a client's (ME's) perspective however, it is difficult to ensure the security of the applet by examining the source code. Therefore, most applets are signed by Certificate Authorities (CAs) (See the section on Public Key Infrastructure (PKI)). Before executing the applet, the subscriber can be informed of the CA which has signed the applet. If the subscriber trusts that CA, they can allow the applet to be executed on their ME otherwise they can block the execution of the applet.

6.4 Security in 3G TWNs

The UMTS security architecture was designed using the GSM security as the starting point. The reason behind doing so was to adopt the GSM security features that have proved to be robust and redesign the features that have been found to be weak. Another reason for doing so was to ensure interoperability between GSM and Universal Mobile Telecommunications System (UMTS) elements.

6.4.1 Anonymity in UMTS

UMTS anonymity builds on the concept of TMSI introduced by GSM (see Section 6.2.1). To avoid subscriber traceability, which may lead to the compromise of subscriber identity, the subscriber should not be identified for a long period by means of the same temporary identity. To achieve this, the UMTS architecture provides provisions for encrypting any signaling or subscriber data that might reveal the subscriber's identity.

Note that we seemingly have a chicken and egg situation here. As we discussed in Section 6.2.1, one of the first things that the ME has to do is to identify itself (its IMSI) to the network. On the other hand, the TMSI allocation procedure (initiated by the VSC/MLR) should be performed after the initiation of ciphering to ensure that the TMSI (and hence the subscriber identity) is not vulnerable to eavesdropping. The problem is that ciphering cannot start unless the CK has been established between the UMTS Subscriber Identity Module (USIM) and the network and the CK cannot be established unless the network first identifies the subscriber using its IMSI. The problem is not as complicated as it appears though. Let's dig a little deeper.

Recall from Section 6.2.1 that the TMSI has only local significance; in other words, the TMSI is allocated by the VLR/MSC and the IMSI-TMSI mapping is maintained in the VLR/MSC. When the subscriber roams into the coverage area of another VLR/MSC (hereafter referred to as VLRn), it continues to identify itself with the TMSI that it was allocated by the previous VLR/MSC (hereafter referred to as VLRo). Obviously VLRn does not recognize this TMSI, since it was allocated by VLRo. In the UMTS architecture, VLRn should request VLRo to get the IMSI corresponding to this TMSI. If VLRn cannot retrieve this information from VLRo, only then should VLRn request the subscriber to identify itself by its IMSI.

The bottom line is that most times, VLRn can determine the IMSI of a subscriber without the ME actually having to transmit the IMSI over the air interface. Instead the ME can identify itself using the TMSI that is already assigned to it. The AKA procedure can then be carried out from this point on.[14] At the completion of the AKA procedure, the CK has been established between the USIM and the network and the VLR/MSC can therefore assign a new TMSI to the ME while ensuring that it is encrypted. From this point on, the TMSI can be safely used by the network and the USIM to identify the subscriber.

Besides the IMSI and the TMSI which can cause a subscriber's identity to be compromised, there is another identity in the UMTS security architecture that can be exploited to trace a subscriber. This is the Sequence Number (SQN) which is used by the ME to authenticate the network. The reason why the SQN can also be used to identify the subscriber is that the network maintains a per-subscriber SQN which

[14] Or VLRn can decide to use a previously existing set of keys.

is incremented sequentially. It is therefore necessary to encrypt the SQN to protect against subscriber traceability. As explained in Section 6.4.2 the authentication process (also known as the Authentication and Key Agreement (AKA)[15] process) in UMTS establishes various keys and one of these is the anonymity key (AK). Just like the CK and IK, the AK is established or derived independently at the USIM and the VLR/MSC without ever being transmitted over the air. In other words, the AK is known only to the USIM and the VLR/MSC. The use of the AK is to protect the Sequence Number (SQN) from eavesdropping.

6.4.2 Key Establishment in UMTS

Just like GSM, there is no key-establishment protocol in UMTS and just like GSM it uses a 128-bit preshared secret key (K_i) between the UMTS Subscriber Identity Module (USIM) and the authentication center (AuC) which forms the basis for all security in UMTS.

6.4.3 Authentication in UMTS

The authentication model in UMTS follows closely from the GSM authentication model but there is one significant difference. The authentication procedure is mutual; that is, the network authenticates the subscriber (USIM) and the subscriber (USIM) authenticates the network. This is unlike GSM where there is no provision for the subscriber to authenticate the validity of the network.

Figure 6.10a shows the authentication procedure in UMTS networks. The process starts when a subscriber (USIM) first sends a sign-on message to the base station it wants to connect to (not shown in Figure 6.10a). The base station then contacts the VLR/MSC (via the RNC) to decide whether or not to allow the USIM access to the network. In order to make this decision, the MSC asks the HLR to provide it with a set of authentication vectors. The authentication vector is the equivalent of the security triplets in GSM. The UMTS authentication vector is actually a security quintet[16] which consists of five numbers: RAND (a 128-bit random number), XRES (the 32-bit expected signed response to the RAND), CK (a 128-bit session cipher or encryption key), IK (a 128-bit integrity key) and AUTN (a 128-bit network authentication token).

When the HLR receives a request for generating authentication vectors from a MSC/VLR, it first generates a random number, RAND and a sequence number, SQN. The

[15] Authentication and key agreement.

[16] Recall from Section 5.1.2 that a GSM security triplet consists of three numbers: RAND (a 128-bit random number), SRES (a 32-bit signed response to the RAND generated using the preshared K_i) and a session key K_c (an encryption key generated using K_i).

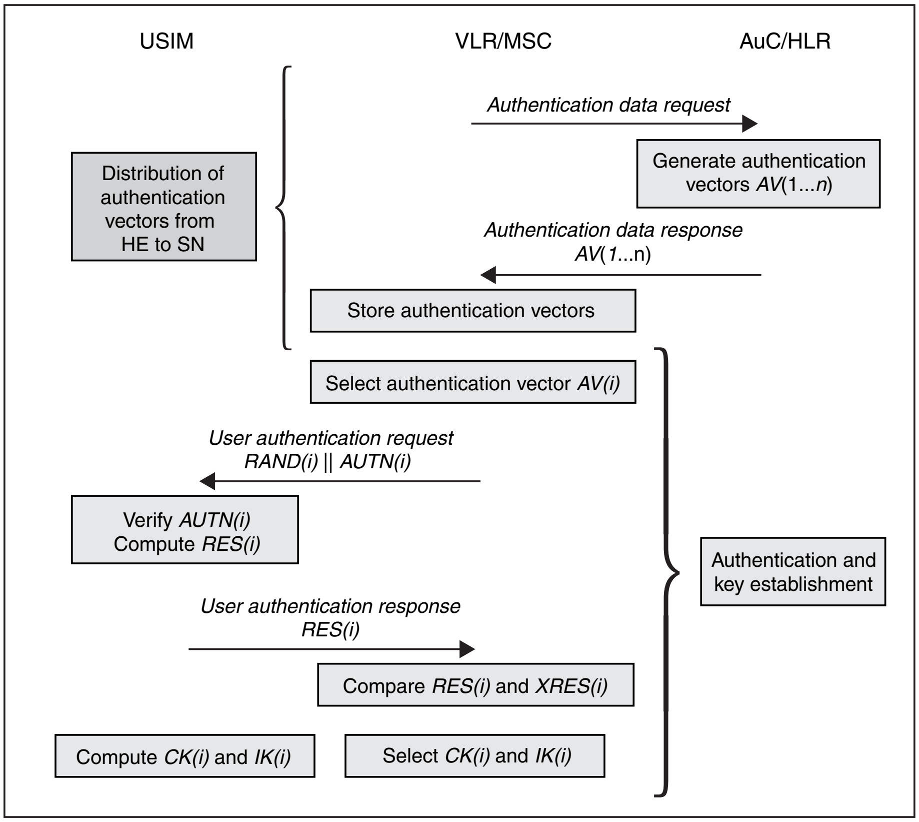

USIM
VLR/MSC
AuC/HLR
Authentication data request
Generate authentication vectors AV(1...n)
Distribution of authentication vectors from HE to SN
Authentication data response AV(1...n)
Store authentication vectors
Select authentication vector AV(i)
User authentication request RAND(i) || AUTN(i)
Verify AUTN(i) Compute RES(i)
Authentication and key establishment
User authentication response RES(i)
Compare RES(i) and XRES(i)
Compute CK(i) and IK(i)
Select CK(i) and IK(i)

Figure 6.10a: UMTS Authentication

HLR then requests the AuC to supply the preshared secret K_i corresponding to this USIM. The RAND, SQN, K_i and the Authentication Management Field (AMF) serve as the input to the five functions (f1 – f5) and generate the security quintet. We tabulate below all the relevant entities involved in the process.

RAND: Random challenge generated by AuC.

XRES: $f2_K$(RAND): Expected subscriber RESponse as computed by AuC.

CK: $f3_K$(RAND): Cipher Key used for encrypting data and signaling messages.

IK: $f4_K$(RAND): Integrity Key.

AK: $f5_K$(RAND): Anonymity Key.

SQN: Sequence Number.

AMF: Authentication Management Field.

MAC: $f1_K$(SQN || RAND || AMF): Message Authentication Code.

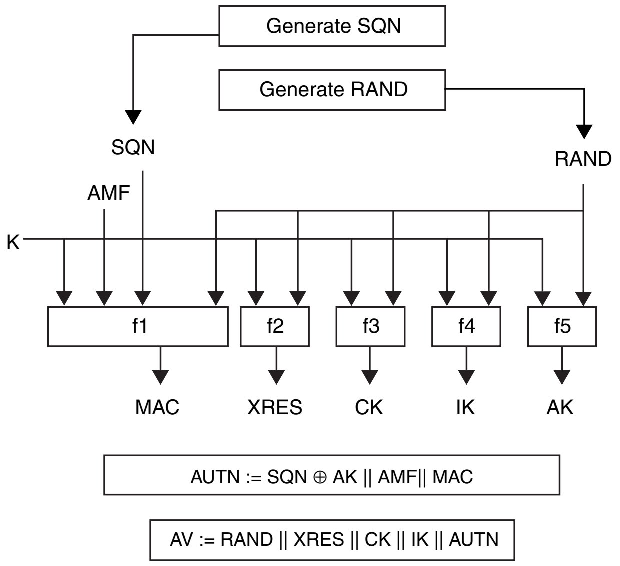

Generate SQN
Generate RAND
SQN
RAND
AMF
K
f1
f2
f3
f4
f5
MAC
XRES
CK
IK
AK
AUTN := SQN ⊕ AK || AMF|| MAC
AV := RAND || XRES || CK || IK || AUTN

Figure 6.10b: UMTS Authentication Vector Generation

AUTN: SQN (+) AK ‖ AMF ‖ MAC: Network Authentication Token.

Security Quintet: (RAND, XRES, CK, IK, AUTN).

When the VLR/MSC receives these set of authentication vectors, it selects the first[17] authentication vector and stores the rest for later use. The VLR/MSC then sends the RAND and the AUTN from the selected authentication vector to the USIM. The RAND serves as a challenge. When the USIM receives these, it carries out the procedure shown in Figure 6.11.

Note that the procedure shown in Figure 6.11 requires only three inputs: AUTN, RAND and K_i. The former two entities are received from the VLR/MSC and K_i is already stored in the USIM. Note also that AUTN basically consists of SQN (+) AK ‖ AMF ‖ MAC which are used as shown in Figure 6.11. Once the USIM has calculated all the entities, it verifies that the MAC received in the AUTN and the XMAC calculated by the USIM match. It also verifies that the SQN obtained from the network's message is in the correct range. Once the USIM has verified these matches, the USIM has authenticated the network since a rogue network can't generate a valid SQN[18].

[17] Note the importance of sequence. In the GSM security architecture, the MSC/VLR selects any one of the five security triplets.

[18] Recall from Section 6.4.1 that the network maintains a per-subscriber SQN which is incremented sequentially each time the ME and the network carry out the authentication process. Also, this SQN can be kept secret by using the AK.

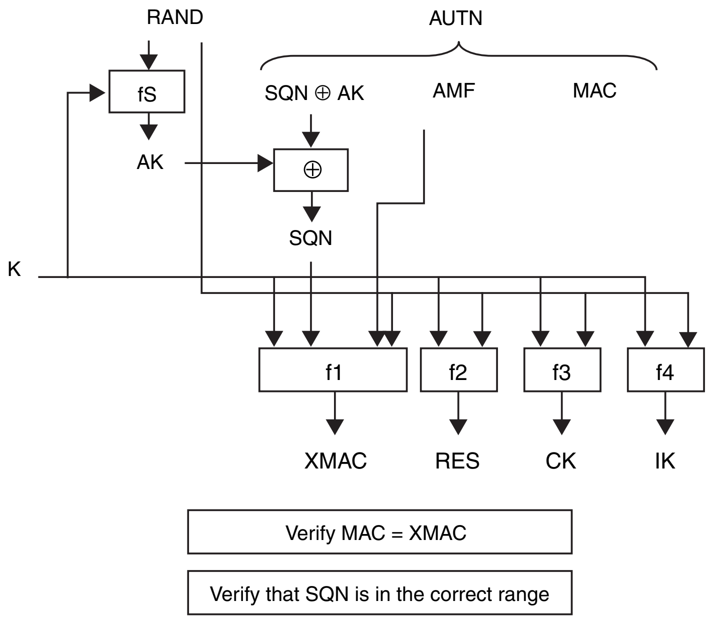


Figure 6.11: UMTS Response Generation at USIM

At this point, one half of the authentication process is now complete. What is now left is for the network to authenticate the USIM. To complete this process, the USIM sends back the RES to the network. Note that the RES is generated from the RAND using the preshared secret key that it stores, K_i and the function f2; i.e., RES = $f2_K$(RAND). The RES is therefore the response to the challenge (RAND). When the VLR/MSC receives the RES from the USIM, it compares it with the XRES in the corresponding authentication vector that it received from the HLR. If the two match, the network (VLR/MSC) has successfully authenticated the USIM and allows it to access network services.

At this point, the mutual authentication process has completed and the following keys have been established between the network and the USIM: CK, IK and AK. These keys now form the basis of providing confidentiality, integrity and anonymity in the UMTS architecture.

Recall from Section 6.2.3 that GSM left the choice of the authentication protocol to the service provider. Most service providers used the COMP128 algorithm for this purpose. UMTS follows the same philosophy. It leaves the choice of the authentication protocol to the service provider but does provide an example algorithm, MILENAGE, that service providers may use.

6.4.4 Confidentiality in UMTS

The GSM encryption algorithm was A5, which used a 64-bit session key K_c. The UMTS encryption algorithm is known as KASUMI and uses a 128-bit session key CK. The KASUMI algorithm is more secure than A5 and one of the reasons for this is simply the use of longer keys for encryption. Figure 6.12 shows the encryption process used in UMTS networks.

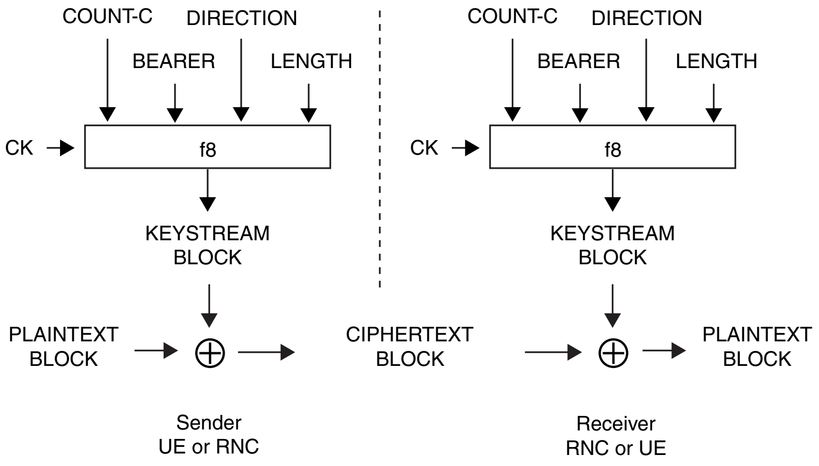


Figure 6.12: UMTS Encryption

Let's take a look at the input values used for encryption. First there is the 128-bit ciphering key (also known as encryption key), CK which has been established at the USIM and at the VLR/MSC as a result of the authentication process. Second, there is the 32-bit COUNT-C which is a ciphering sequence number which is updated sequentially for each plaintext block. Third, there is the 5-bit BEARER which is a unique identifier for the bearer channel (the channel number which is used for carrying end-user's traffic) in use. Fourth, there is the 1-bit DIRECTION value which indicates the direction of transmission (uplink or downlink). Finally, there is the 16-bit LENGTH which indicates the length of the key-stream block. All these values are input into the f8 encryption algorithm[19] to generate a key stream. This key stream is XORed with the plaintext block to generate the ciphertext block. At the receiving end, the same process is repeated except that the generated key stream is XORed with the received ciphertext to get back the plaintext.

[19] Just like in GSM, "f8" is a label for an algorithm rather than an algorithm itself. One oft-used f8 algorithm is the Kasumi algorithm.

Besides the use of increased key lengths, there is another significant improvement in UMTS security architecture over the GSM security architecture. Recall from Section 6.2.2 that the GSM confidentiality was limited to securing the link between the ME and the BTS. This made the link between the BTS and the BSC (usually a microwave link) unsecure. The UMTS security architecture extends the encrypted interface from the BTS back to the RNC (the UMTS equivalent of BSC). Since the traffic between the USIM and RNC is encrypted, this protects not only the wireless interface between the USIM and base station but also the interface between the base station and the RNC, thus closing one of the loopholes of GSM security. One last thing to note is that the encryption in the UMTS security architecture is applied to all subscriber traffic as well as signaling messages.

6.4.5 Integrity Protection in UMTS

As we discussed in Section 6.2.5, one of the gaping loopholes in the GSM security architecture was the absence of an integrity protection mechanism. UMTS attempts to solve this problem using the integrity key IK derived using the authentication process as described in Section 6.4.2.

The UMTS integrity mechanism is shown in Figure 6.13. Let's look at the input values required for the integrity mechanism. First, there is the 128-bit integrity key, IK, which is established as a part of the UMTS authentication process. Second, there is the 32-bit integrity sequence number which is updated sequentially for each plaintext block that is integrity protected. Third, there is the message itself which needs to be integrity protected. Fourth, there is the DIRECTION bit (uplink or downlink). Finally, there is the 32-bit FRESH which is a per-connection nonce. All these values are input to the f9 algorithm and the output is a 32-bit MAC-I (message authentication code). This MAC is attached to the message by the sender. At the receiving end, the same

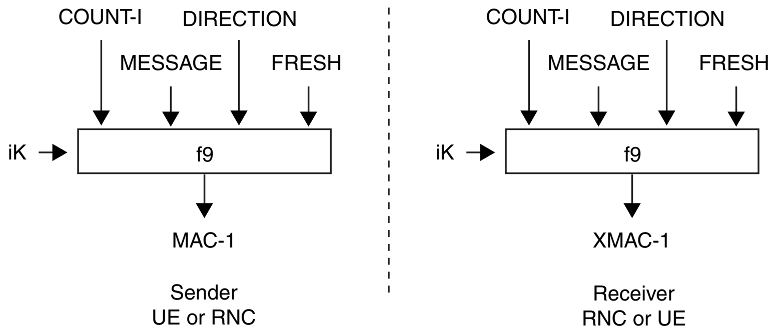


Figure 6.13: UMTS Message Integrity

process is repeated to calculate the XMAC-I. The receiver then compares the computed XMAC to the received MAC, so the receiver can deduce that the message was not tampered with. Thus we have integrity protected the message. The integrity protection mechanism just described is applied to all but a specifically excluded set of signaling messages.

Ideally, the integrity protection mechanism should be used for protecting user traffic too. However, integrity protecting each packet involves a lot of overhead in terms of processing and bandwidth, but let's take a step back and think about it: what is it that we are trying to achieve by integrity protecting user traffic? The aim is to ensure that the information content of the voice conversation is not modified; in other words, words are not inserted, deleted or modified in the conversation. However, ensuring this does not really require that each voice packet be integrity protected, since inserting, deleting or modifying a word in a conversation (without the user realizing) would affect voice samples spanning several packets.[20] Almost invariably, inserting, deleting or modifying words in a conversation without the user realizing this, would lead to a change in the number of packets. It is therefore usually sufficient to integrity protect the number of user packets in a conversation. This is the compromise that UMTS uses. The RNC monitors the sequence numbers that are used for ciphering voice packets related to each radio bearer (channel). Periodically, the RNC may send a message containing these sequence numbers (which reflect the amount of data sent and received on each active radio bearer) to the ME. Obviously, this message itself is integrity protected. On receiving this message, the ME can verify that the value received in this message matches the counter values that the ME maintains locally. This provides integrity protection of the user traffic.

6.4.6 Putting the Pieces Together

The UMTS security architecture is huge, complex and entails a lot of features. We have seen individual pieces of this security architecture in the last few sections. In this section, we try to put all the pieces together to get a complete picture.

Figure 6.14 shows the overview of the UMTS security process. The process starts when the ME first starts the Layer 2 connection (RRC layer connection) procedure. Note that the Layer 2 connection refers to the connection between the MS and the RNC. The message exchange between the ME and the VLR/MSC is a Layer 3 level

[20] Voice Packets basically consist of samples of digitized voice. Typical voice calls over TWNs usually use a rate of 10–20 Kbps to digitize voice. Therefore a single spoken word would usually span several packets.

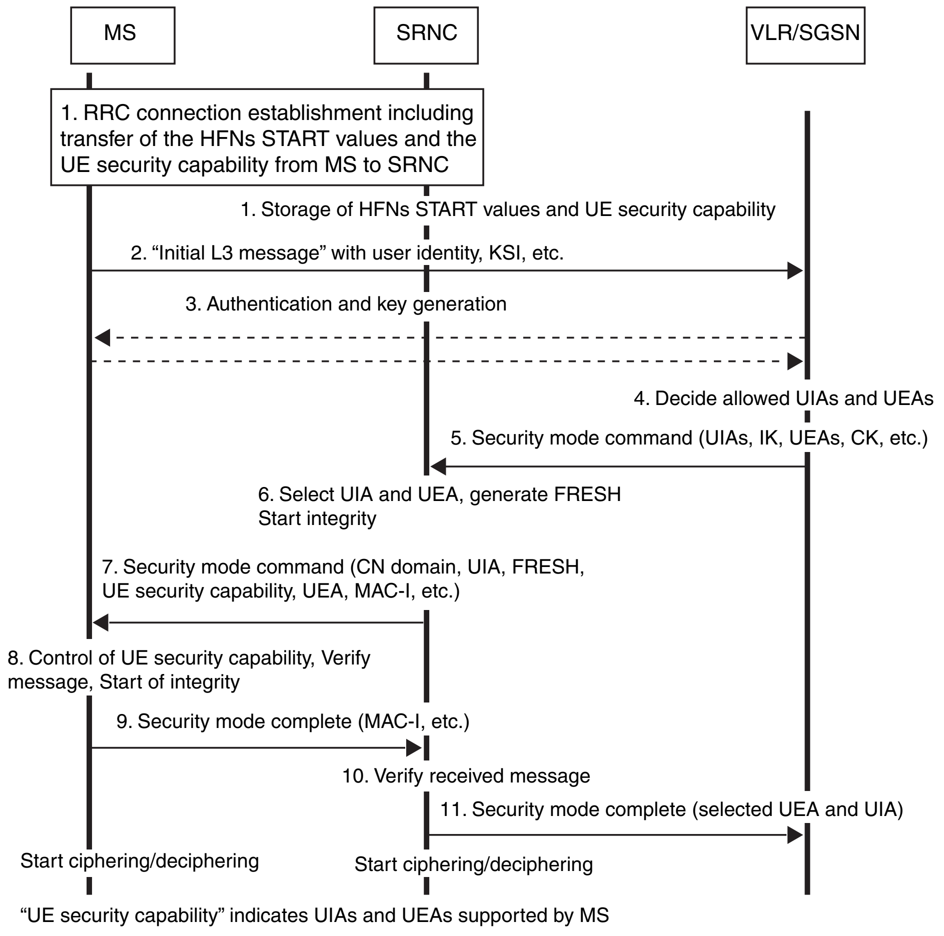

MS
SRNC
VLR/SGSN
1. RRC connection establishment including transfer of the HFNs START values and the UE security capability from MS to SRNC
1. Storage of HFNs START values and UE security capability
2. "Initial L3 message" with user identity, KSI, etc.
3. Authentication and key generation
4. Decide allowed UIAs and UEAs
5. Security mode command (UIAs, IK, UEAs, CK, etc.)
6. Select UIA and UEA, generate FRESH
Start integrity
7. Security mode command (CN domain, UIA, FRESH, UE security capability, UEA, MAC-I, etc.)
8. Control of UE security capability, Verify message, Start of integrity
9. Security mode complete (MAC-I, etc.)
10. Verify received message
11. Security mode complete (selected UEA and UIA)
Start ciphering/deciphering
Start ciphering/deciphering
"UE security capability" indicates UIAs and UEAs supported by MS

Figure 6.14: UMTS Security—*Overview*

connection. In any case, as part of the RRC layer connection procedure, the MS sends, among other things, its list of security capabilities: User Encryption Algorithms (UEAs), User Integrity Algorithms (UIAs) and so on to the RNC. The RNC stores these subscriber security capabilities.

When the ME sends its first Layer 3 Connection message to the VLR/MSC, it includes in this message the subscriber identity (either the TMSI or the IMSI) and Key Set Identifier (KSI). The KSI identifies the set of keys (CK, IK, and so on) that were last established between this ME and this CN domain.[21] Note that the first Layer 3 message can be any one of a set of possible initial Layer 3 messages (location update

[21] Usually a service provider's domain.

request, routing update request, attach request and so on). The important thing is that whichever Layer 3 is sent first,[22] it will carry the subscriber identity and the KSI.

Once the VLR/MSC receives the subscriber identity and the KSI, it may decide to start a new AKA procedure or it may decide to continue using the keys that it already shares with the ME from last time. This decision is left to the VLR/MSC. In other words, the service provider is free to control when (and how frequently) it authenticates the identity of the subscriber and establishes a new key set. In either case, at this point, both the ME (the USIM to be precise) and the VLR/MSC trust each other's identity and have agreed upon the set of keys (CK, IK) that they will use.

Note however that the security algorithms to be used have not been agreed on. This is the next step. Towards this end, the VLR/MSC sends a list of the encryption and integrity algorithms that this subscriber is allowed to use to the RNC. The RNC now has the set of UEAs and the UIAs that the ME wishes to use and also the set of encryption and integrity algorithms that the VLR/MSC wishes to use. The RNC takes the intersection of these two sets and determines an encryption and integrity algorithm that is most acceptable to the ME and the VLR/MSC. In other words, the RNC has now decided upon the encryption and the integrity algorithm that would be used between the ME and the VLR/MSC. It does however need to pass this information on to the ME and the VLR/MSC.

The RNC sends the selected algorithm information in the next message. Also included in this message is the full set of security capabilities that the MS had sent to the RNC during RRC connection establishment time. Sending the UEAs and UIAs that the RNC received from the MS back to the MS may seem redundant at first. Realize however, that this is the first message that is integrity protected; in other words, no message sent over the air interface before this message was integrity protected.[23] This means that the set of security capabilities that the MS had sent to the RNC as part of RRC connection establishment procedure was not integrity protected. Therefore, a malicious eavesdropper could have modified this message to change the set of security capabilities that the MS offers to the network thus forcing the MS and the network to use weaker encryption and integrity algorithms. Such an attack is known as the bidding down attack. To prevent against such an attack, the RNC returns the full set of security capabilities that the MS had sent to the RNC in an integrity protected message.

[22] This will depend on the context in which it is sent.
[23] But all messages from now on are integrity protected.

On receiving this message, the MS ensures the integrity of this message by calculating the XMAC on this message and comparing it with the MAC received in the message, thus verifying that the message has not been modified by a malicious eavesdropper. Next, the MS verifies that the set of capabilities in this message is the same as the set of capabilities that the MS had sent to the RNC during the RRC connection setup. Once the MS has verified that the set of security capabilities were not modified, it sends a security mode complete message back to the RNC. Note that this and all subsequent messages from the MS are integrity protected.

At this point the MS has authenticated the network, the network has authenticated the MS and the MS has all the information (algorithms, keys and so on) that it requires to provide confidentiality and integrity protection. However, the VLR/MSC does not have all the information that it requires for this purpose since the encryption and the integrity protection algorithms to be used were determined by the RNC and never conveyed to the VLR/MSC. This is what is done next. The RNC sends a message specifying the encryption and the integrity protection algorithms to be used to the VLR/MSC. From this point on not only are all the messages integrity protected but also encrypted to provide confidentiality.

6.4.7 Network Domain Security

In the last few sections, we have seen how UMTS provides security in the access network. The term *access network* here refers to the connection between the ME and the VLR/MSC. As we discussed in Section 6.2, second generation TWNs were concerned only with securing the wireless access network and the core network was considered to be a secure operating environment. This security in the core network was based on the fact that the core network was accessible to only a relatively small number of well established institutions and it was therefore very difficult for an attacker to get access to this network.

However, this assumption of the core network being inherently secure is no longer valid since with the opening up of telecom regulations all over the world, the number of service providers has grown significantly. This has two important implications. One, there are a lot more institutions that now have access to the core network, thus increasing the probability that the core network may be compromised. Two, since the access networks of these service providers need to communicate with each other (to provide seamless mobility for example), there is a growing need to make this internetwork communication secure.

The ideal solution obviously would have been to secure the core network. The problem is that this was a huge task and beyond the scope of UMTS network designers.

The UMTS designers therefore limited their scope to securing the mobile specific part of the network, which is known as the Mobile Application Part (MAP). To this end, UMTS specifies the MAPSEC protocol, which works at the application layer to protect MAP messages cryptographically.[24] Figure 6.15 shows how MAPSEC is used for protecting MAP messages being exchanged between two networks.

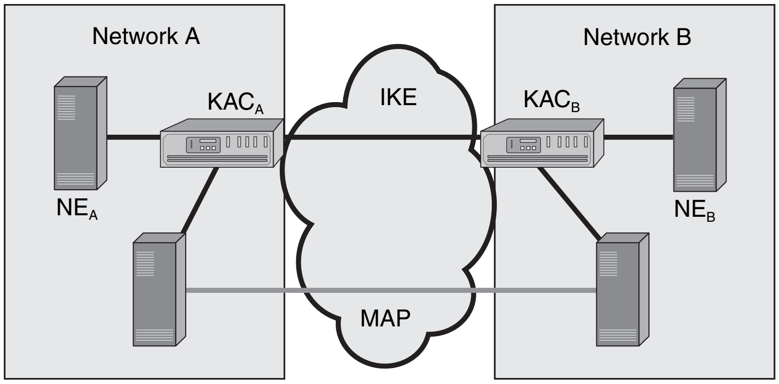


Figure 6.15: MAPSEC

The Key Administration Center (KAC) is a new entity introduced in the system architecture by MAPSEC. Each network which wishes to use MAPSEC has a KAC. The purpose of KAC in Network A is to establish a Security Association (SA) with the KAC in Network B. The term security association refers to the set of security algorithms, keys, key lifetimes and so on that the two networks will use to secure MAP messages that they exchange. To establish a SA, the KACs use the Internet Key Exchange (IKE) protocol. Once the SAs have been established, the KAC distributes this information to its Network Elements (NE). The network elements then use these SAs to protect the MAP messages. MAPSEC allows for three modes of protection: no protection, integrity protection only and integrity with confidentiality.

As you will have noticed, the design of the MAPSEC protocol is strongly influenced by the IPSec protocol. This influence comes across in the use of IKE for key establishment and the use of SAs, among other things. There is an important reason for this. The exploding growth in data networks in the last few years has led to the convergence of voice and data networks and the boundaries between these two networks are fast disappearing. 2.5G TWNs marked the integration of data networks with TWNs. 3G networks are expected to be even more closely tied to IP-based

[24] Some MAP messages are still sent in plaintext without any protection, to avoid performance penalties.

networks. This means that replacing SS7 signaling with IP-based signaling (like SIP) is extremely likely. Therefore, the UMTS network designers provided a method not only for securing MAP in SS7 networks (MAPSEC) but also for using MAP over IP-based networks which may be protected by the already well-established IPSec protocol. Keeping this convergence in mind, the UMTS network designers tried to model MAPSEC along the IPSec lines.

Figure 6.16 shows how network domain security is achieved for IP-based control messages (out of which MAP messages may just be one type of messages being protected). Note that Figure 6.16 resembles Figure 6.15 closely. The only difference seems to be that the KAC has been replaced by another entity known as the Security Gateway (SEG). Like the KAC, the SEG in Network A establishes SAs with its peer in Network B. However, unlike the KAC, the SEG does not distribute the SAs to its network elements. Instead, it maintains a database of established SAs and a database of security policies which specify how and when the SAs are to be used. When the network elements in Network A want to send control messages (like MAP messages) to their peers in Network B, they send the message to the SEG. It is the SEG which is then responsible for protecting the message in accordance with the policy using the established SA. In other words, the SEG is responsible not only for establishing the SAs but also for using them to protect control messages.

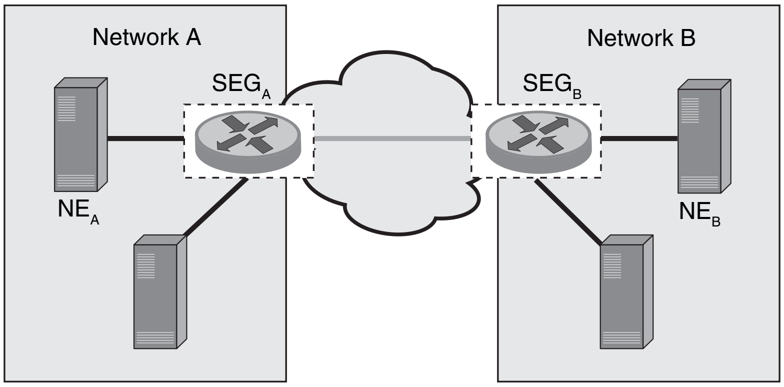


Figure 6.16: MAP Over IP-based Networks

6.5 Summary

From the rudimentary ESN in first generation TWNs to the provisions for confidentiality, integration, mutual authentication, and anonymity in the UMTS networks, security in TWNs has come a long way. This improvement in security can be attributed to many factors. Probably the most important among these are the demand for security from subscribers and technological enhancements in cryptography. There is also however another important factor which has made this possible and this is the Moore's Law. Remember that security always comes at a cost. Without the significant growth in processing power per square millimeter, it would probably have been impossible to provide the kind of security that TWN subscribers have come to expect. In the near future, the convergence of TWNs with the Internet will surely bring new challenges. What is important is that we be willing to admit the loopholes and then fix them.

CHAPTER 7

Security in Wireless Local Area Networks

7.1 Introduction

The 802.11 security architecture and protocol is called Wired Equivalent Privacy (WEP). It is responsible for providing authentication, confidentiality and data integrity in 802.11 networks. To understand the nomenclature, realize that 802.11 was designed as a "wireless Ethernet." The aim of the WEP designers was therefore to provide the same degree of security as is available in traditional wired (Ethernet) networks. Did they succeed in achieving this goal?

A few years back, asking that question in the wireless community was a sure-fire way of starting a huge debate. To understand the debate, realize that wired Ethernet[1] (the IEEE 802.3 standard) implements no security mechanism in hardware or software. However, wired Ethernet networks are inherently "secured" since the access to the medium (wires) which carry the data can be restricted or secured. On the other hand, in "wireless Ethernet" (the IEEE 802.11 standard) there is no provision to restrict access to the (wireless) media. So, the debate was over whether the security provided by WEP (the security mechanism specified by 802.11) was comparable to (as secure as) the security provided by restricting access to the physical medium in wired Ethernet. Since this comparison is subjective, it was difficult to answer this question. In the absence of quantitative data for comparison, the debate raged on. However, recent loopholes discovered in WEP have pretty much settled the debate, concluding that WEP fails to achieve its goals.

In this chapter, we look at WEP, why it fails and what is being done to close these loopholes. It is interesting to compare the security architecture in 802.11 with the security architecture in Traditional Wireless Networks (TWNs). Note that both TWNs and 802.11 use the wireless medium only in the access network; that is, the part of the

[1] We use 802.3 as a standard of comparison since it is the most widely deployed LAN standard. The analogy holds true for most other LAN standards—more or less.

network which connects the end-user to the network. This part of the network is also referred to as the last hop of the network. However, there are important architectural differences between TWNs and 802.11.

The aim of TWNs was to allow a wireless subscriber to communicate with any other wireless or wired subscriber anywhere in the world while supporting seamless roaming over large geographical areas. The scope of the TWNs therefore, went beyond the wireless access network and well into the wired network.

On the other hand, the aim of 802.11 is only last-hop wireless connectivity. 802.11 does not deal with end-to-end connectivity. In fact, IP-based data networks (for which 802.11 was initially designed) do not have any concept of end-to-end connectivity and each packet is independently routed. Also, the geographical coverage of the wireless access network in 802.11 is significantly less than the geographical coverage of the wireless access network in TWNs. Finally, 802.11 has only limited support for roaming. For all these reasons, the scope of 802.11 is restricted to the wireless access network only. As we go along in this chapter, it would be helpful to keep these similarities and differences in mind.

7.2 Key Establishment in 802.11

The key establishment protocol of 802.11 is very simple to describe—there is none. 802.11 relies on "preshared" keys between the mobile nodes or stations (henceforth Stations (STAs)) and the Access Points (APs). It does not specify how the keys are established and assumes that this is achieved in some "out-of-band" fashion. In other words, key establishment is outside the scope of WEP.

7.2.1 What's Wrong?

As we saw in Chapter 2, key establishment is one of the toughest problems in network security. By not specifying a key establishment protocol, it seems that the 802.11 designers were side-stepping the issue. To be fair to 802.11 designers, they did a pretty good job with the standard. The widespread acceptance of this technology is a testament to this. In retrospect, security was one of the issues where the standard did have many loopholes, but then again everyone has perfect vision in hindsight. Back to our issue, the absence of any key management protocol led to multiple problems as we discuss below.

1. In the absence of any key management protocol, real life deployment of 802.11 networks ended up using manual configuration of keys into all STAs and the AP that wish to form a Basic Service Set (BSS).

2. Manual intervention meant that this approach was open to manual error.
3. Most people cannot be expected to choose a "strong" key. In fact, most humans would probably choose a key which is easy to remember. A quick survey of the 802.11 networks that I had access to shows that people use keys like "abcd1234" or "12345678" or "22222222" and so on. These keys, being alphanumeric in nature, are easy to guess and do not exploit the whole key space.
4. There is no way for each STA to be assigned a unique key. Instead, all STAs and the AP are configured with the same key. As we will see in Section 7.4.4, this means that the AP has no way of uniquely identifying a STA in a secure fashion. Instead, the STAs are divided into two groups. Group One consists of stations that are allowed access to the network, and Group Two consists of all other stations (that is, STAs which are not allowed to access the network). Stations in Group One share a secret key which stations in Group Two don't know.
5. To be fair, 802.11 does allow each STA (and AP) in a BSS to be configured with four different keys. Each STA can use any one of the four keys when establishing a connection with the AP. This feature may therefore be used to divide STAs in a BSS into four groups if each group uses one of these keys. This allows the AP a little finer control over reliable STA recognition.
6. In practice, most real life deployments of 802.11 use the same key across BSSs over the whole extended service set (ESS).[2] This makes roaming easier and faster, since an ESS has many more STAs than a BSS. In terms of key usage, this means that the same key is shared by even more STAs. Besides being a security loophole to authentication (see Section 7.4.4), this higher exposure makes the key more susceptible to compromise.

7.3 Anonymity in 802.11

We saw that subscriber anonymity was a major concern in TWNs. Recall that TWNs evolved from the voice world (the PSTN). In data networks (a large percentage of which use IP as the underlying technology), subscriber anonymity is not such a major concern. To understand why this is so, we need to understand some of the underlying architectural differences between TWNs and IP-based data networks. As we saw

[2] Recall that an ESS is a set of APs connected by a distribution system (like Ethernet).

in Chapter 4, TWNs use IMSI for call routing. The corresponding role in IP-based networks is fulfilled by the IP address. However, unlike the IMSI, the IP address is not permanently mapped to a subscriber. In other words, given the IMSI, it is trivial to determine the identity of the subscriber. However, given the IP address, it is extremely difficult to determine the identity of the subscriber. This difficulty arises because of two reasons. First, IP addresses are dynamically assigned using protocols like DHCP; in other words, the IP address assigned to a subscriber can change over time.

Second, the widespread use of Network Address Translation (NAT) adds another layer of identity protection. NAT was introduced to deal with the shortage of IP addresses.[3] It provides IP-level access between hosts at a site (local area network (LAN)) and the rest of the Internet without requiring each host at the site to have a globally unique IP address. NAT achieves this by requiring the site to have a single connection to the global Internet and at least one globally valid IP address (hereafter referred to as GIP). The address GIP is assigned to the NAT translator (also known as NAT box), which is basically a router that connects the site to the Internet. All datagrams coming into and going out of the site must pass through the NAT box. The NAT box replaces the source address in each outgoing datagram with GIP and the destination address in each incoming datagram with the private address of the correct host. From the view of any host external to the site (LAN), all datagrams come from the same GIP (the one assigned to the NAT box). There is no way for an external host to determine which of the many hosts at a site a datagram came from. Thus, the usage of NAT adds another layer of identity protection in IP networks.

7.4 Authentication in 802.11

Before we start discussing the details of authentication in 802.11 networks, recall that the concept of authentication and access control are very closely linked. To be precise, one of the primary uses of authentication is to control access to the network. Now, think of what happens when a station wants to connect to a LAN. In the wired world, this is a simple operation. The station uses a cable to plug into an Ethernet jack, and it is connected to the network. Even if the network does not explicitly authenticate the station, obtaining physical access to the network provides at least some basic access control if we assume that access to the physical medium is protected. In the wireless world, this physical-access-authentication disappears.

[3] To be accurate, the shortage of IPv4 addresses. There are more than enough IPv6 addresses available but the deployment of IPv6 has not caught on as fast as its proponents would have liked.

For a station to "connect to" or associate with a wireless local area network (WLAN), the network-joining operation becomes much more complicated. First, the station must find out which networks it currently has access to. Then, the network must authenticate the station and the station must authenticate the network. Only after this authentication is complete can the station *connect to* or associate with the network (via the AP). Let us go over this process in detail.

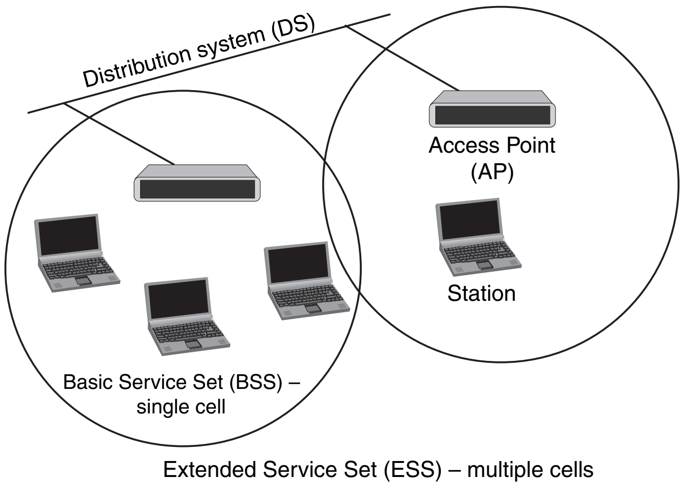

Distribution system (DS)
Access Point (AP)
Station
Basic Service Set (BSS) – single cell
Extended Service Set (ESS) – multiple cells

Figure 7.1: 802.11 System Overview

Access points (APs) in an 802.11 network periodically broadcast beacons. Beacons are management frames which announce the existence of a network. They are used by the APs to allow stations to find and identify a network. Each beacon contains a Service Set Identifier (SSID), also called the *network name*, which uniquely identifies an ESS. When an STA wants to access a network, it has two options: passive scan and active scan. In the former case, it can scan the channels (the frequency spectrum) trying to find beacon advertisements from APs in the area. In the latter case, the station sends probe-requests (either to a particular SSID or with the SSID set to 0) over all the channels one-by-one. A particular SSID indicates that the station is looking for a particular network. If the concerned AP receives the probe, it responds with a probe-response. A SSID of 0 indicates that the station is looking to join any network it can access. All APs which receive this probe-request and which want this particular station to join their network, reply back with a probe-response. In either case, a station finds out which network(s) it can join.

Next, the station has to choose a network it wishes to join. This decision can be left to the user or the software can make this decision based on signal strengths and other criteria. Once a station has decided that it wants to join a particular network, the authentication process starts. 802.11 provides for two forms of authentication: Open System Authentication (OSA) and Shared Key Authentication (SKA). Which authentication is to be used for a particular transaction needs to be agreed upon by both the STA and the network. The STA proposes the authentication scheme it wishes to use in its authentication request message. The network may then accept or reject this proposal in its authentication response message depending on how the network administrator has set up the security requirements of the network.

7.4.1 Open System Authentication

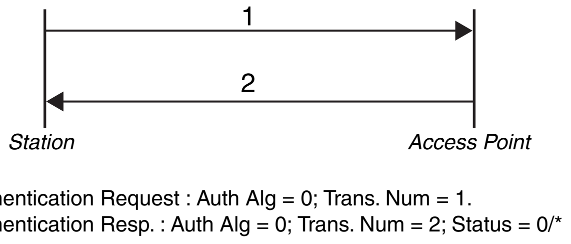

1
2
Station
Access Point
1. Authentication Request : Auth Alg = 0; Trans. Num = 1.
2. Authentication Resp. : Auth Alg = 0; Trans. Num = 2; Status = 0/*

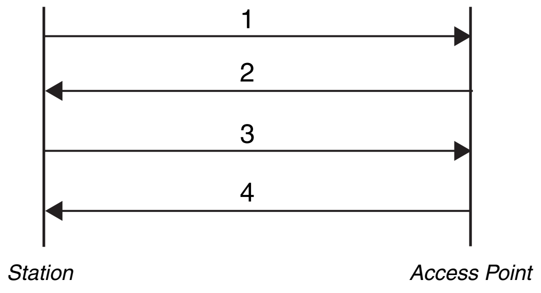
Figure 7.2: 802.11 OSA

This is the default authentication algorithm used by 802.11. Here is how it works. Any station which wants to join a network sends an authentication request to the appropriate AP. The authentication request contains the authentication algorithm that the station the wishes to use (0 in case of OSA). The AP replies back with an authentication response thus authenticating the station to join the network[4] if it has been configured to accept OSA as a valid authentication scheme. In other words, the AP does not do any checks on the identity of the station and allows any and all stations to join the network. OSA is exactly what its name suggests: open system authentication. The AP (network) allows any station (that wishes to join) to join the network. Using OSA therefore means using no authentication at all.

It is important to note here that the AP can enforce the use of authentication. If a station sends an authentication request requesting to use OSA, the AP may deny the station access to the network if the AP is configured to enforce SKA on all stations.

[4] The authentication request from the station may be denied by the AP for reasons other than authentication failure, in which case the status field will be nonzero.

7.4.2 Shared Key Authentication



1. Authentication Request : Auth Alg = 1; Trans. Num = 1.
2. Authentication Resp : Auth Alg = 1; Trans. Num = 2; Data = 128-byte random number.
3. Authentication Resp : Auth Alg = 1; Trans. Num = 3; Data = Encrypted (128-byte number rcvd in.
4. Authentication Resp : Auth Alg = 1; Trans. Num = 4; Status = 0/*.

Figure 7.3: 802.11 SKA

SKA is based on the challenge-response system described in Section 1.2.3. SKA divides stations into two groups. Group One consists of stations that are allowed access to the network and Group Two consists of all other stations. Stations in Group One share a secret key which stations in Group Two don't know. By using SKA, we can ensure that only stations belonging to Group One are allowed to join the network.

Using SKA requires 1) that the station and the AP be capable of using WEP and 2) that the station and the AP have a preshared key. The second requirement means that a shared key must be distributed to all stations that are allowed to join the network before attempting authentication. How this is done is not specified in the 802.11 standard. Figure 7.3 explains how SKA works in detail.

When a station wants to join a network, it sends an authentication request to the appropriate AP which contains the authentication algorithm it wishes to use (1 in case of SKA). On receiving this request, the AP sends an authentication response back to the station. This authentication response contains a challenge-text. The challenge text is a 128-byte number generated by the pseudorandom-number-generator (also used in WEP) using the preshared secret key and a random Initialization Vector (IV). When the station receives this random number (the challenge), it encrypts the random number using WEP[5] and its own IV to generate a response to the challenge. Note that the IV that the station uses for encrypting the challenge is different from (and independent of) the IV that the AP used for generating the random number. After encrypting

[5] WEP is described in Section 7.5.

the challenge, the station sends the encrypted challenge and the IV it used for encryption back to the AP as the response to the challenge. On receiving the response, the AP decrypts the response using the preshared keys and the IV that it receives as part of the response. The AP compares the decrypted message with the challenge it sent to the station. If these are the same, the AP concludes that the station wishing to join the network is one of the stations which knows the secret key and therefore the AP authenticates the station to join the network.

The SKA mechanism allows an AP to verify that a station is one of a select group of stations. The AP verifies this by ensuring that the station knows a secret. This secret is the preshared key. If a station does not know the key, it will not be able to respond correctly to the challenge. Thus, the strength of SKA lies in keeping the shared key a secret.

7.4.3 Authentication and Handoffs

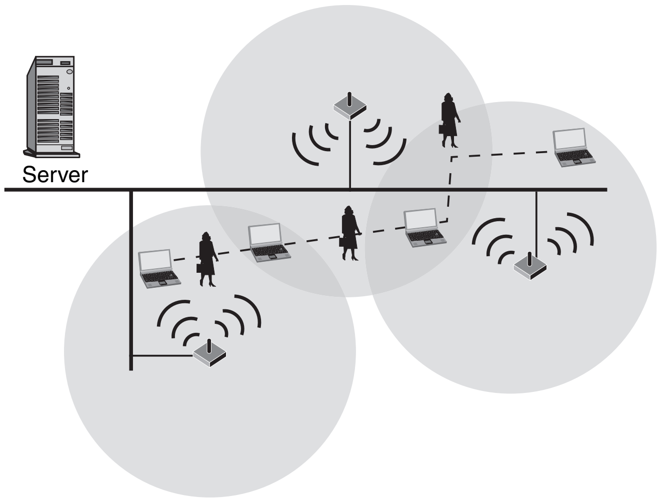


Figure 7.4: 802.11 Handoffs and Security

If a station is mobile while accessing the network, it may leave the range of one AP and enter into the range of another AP. In this section we see how authentication fits in with mobility.

A STA may move inside a BSA (intra-BSA), between two BSAs (inter-BSA) or between two Extended Service Areas (ESAs) (inter-ESAs). In the intra-BSA case, the STA is static for all handoff purposes. Inter-ESA roaming requires support from higher layers (MobileIP for example) since ESAs communicate with each other at Layer 3.

It is the inter-BSA roaming that 802.11 deals with. A STA keeps track of the received signal strength (RSS) of the beacon with which it is associated. When this RSS value falls below a certain threshold, the STA starts to scan for stronger beacon signals available to it using either active or passive scanning. This procedure continues until the RSS of the current beacon returns above the threshold (in which case the STA stops scanning for alternate beacons) or until the RSS of the current beacon falls below the break-off threshold, in which case the STA decides to handoff to the strongest beacon available. When this situation is reached, the STA disconnects from its prior AP and connects to the new AP afresh (just as if had switched on in the BSA of the new AP). In fact, the association with the prior-AP is not "carried-over" or "handed-off" transparently to the new AP: the STA disconnects with the old AP and then connects with the new AP.

To connect to the new AP, the STA starts the connection procedure afresh. This means that the process of associating (and authenticating) to the new AP is the same as it is for a STA that has just powered on in this BSS. In other words, the prior-AP and the post-AP do not co-ordinate among themselves to achieve a handoff.[6] Analysis[7] has shown that authentication delays are the second biggest contributors to handoff times next only to channel scanning/probing time. This re-authentication delay becomes even more of a bottleneck for real time applications like voice. Although this is not exactly a security loophole, it is a "drawback" of using the security.

7.4.4 What's Wrong with 802.11 Authentication?

Authentication mechanisms suggested by 802.11 suffer from many drawbacks. As we saw, 802.11 specifies two modes of authentication—OSA and SKA. OSA provides no authentication and is irrelevant here.

SKA works on a challenge-response system as explained in Section 7.4.2. The AP expects that the challenge it sends to the STA be encrypted using an IV and the pre-shared key. As described in Section 7.2.1, there is no method specified in WEP for each STA to be assigned a unique key. Instead all STAs and the AP in a BSS are configured with the same key. This means that even when an AP authenticates a STA using the

[6] To be accurate, the IEEE 802.11 standard does not specify how the two APs should communicate with each other. There do exist proprietary solutions by various vendors which enable inter-AP communication to improve handoff performance.

[7] An Empirical Analysis of the IEEE 802.11 MAC layer Handoff Process—Mishra et al.

SKA mode, all it ensures is that the STA belongs to a group of STAs which know the preshared key. There is no way for the AP to reliably determine the exact identity of the STA that is trying to authenticate to the network and access it.[8]

To make matters worse, many 802.11 deployments share keys across APs. This increases the size of the group to which a STA can be traced. All STAs sharing a single preshared secret key also makes it very difficult to remove a STA from the allowed set of STAs, since this would involve changing (and redistributing) the shared secret key to all stations.

There is another issue with 802.11 authentication: it is one-way. Even though it provides a mechanism for the AP to authenticate the STA, it has no provision for the STA to be able to authenticate the network. This means that a rogue AP may be able to hijack the STA by establishing a session with it. This is a very plausible scenario given the plummeting cost of APs. Since the STA can never find out that it is communicating with a rogue AP, the rogue AP has access to virtually everything that the STA sends to it.

Finally, SKA is based on WEP, discussed in Section 7.5. It therefore suffers from all the drawbacks that WEP suffers from too. These drawbacks are discussed in Section 7.5.1.

7.4.5 Pseudo-Authentication Schemes

Networks unwilling to use SKA (or networks willing to enhance it) may rely on other authentication schemes. One such scheme allows only stations which know the network's SSID to join the network. This is achieved by having the AP responding to a probe-request from a STA only if the probe request message contains the SSID of the network. This in effect prevents connections from STAs looking for any wild carded SSIDs. From a security perspective, the secret here is the SSID of the network. If a station knows the SSID of the network, it is allowed to join the network. Even though this is a very weak authentication mechanism, it provides some form of protection against casual eavesdroppers from accessing the network. For any serious eavesdropper (hacker), this form of authentication poses minimal challenge since the SSID of the network is often transmitted in the clear (without encryption).

Yet another authentication scheme (sometimes referred to as address filtering) uses the MAC addresses as the secret. The AP maintains a list of MAC addresses of all the STAs that are allowed to connect to the network. This table is then used for admission

[8] MAC addresses can be used for this purpose but they are not cryptographically protected in that it is easy to spoof a MAC address.

control into the network. Only stations with the MAC addresses specified in the table are allowed to connect to the network. When a station tries to access the network via the AP, the AP verifies that the station has a MAC address which belongs to the above mentioned list. Again, even though this scheme provides some protection, it is not a very secure authentication scheme since most wireless access cards used by stations allow the user to change their MAC address via software. Any serious eavesdropper or hacker can find out the MAC address of one of the stations which is allowed access by listening in on the transmissions being carried out by the AP and then change their own MAC address to the determined address.

7.5 Confidentiality in 802.11

WEP uses a preestablished/preshared set of keys. Figure 7.5 shows how WEP is used to encrypt an 802.11 MAC Protocol Data Unit (MPDU). Note that Layer 3 (usually IP) hands over a MAC Service Data Unit (MSDU) to the 802.11 MAC layer. The 802.11 protocol may then fragment the MSDU into multiple MPDUs if so required to use the channel efficiently.

The WEP process can be broken down in to the following steps.

Step 1: Calculate the Integrity Check Value (ICV) over the length of the MPDU and append this 4-byte value to the end of the MPDU. Note that ICV is another name for Message Integrity Check (MIC). We see how this ICV value is generated in Section 7.6.

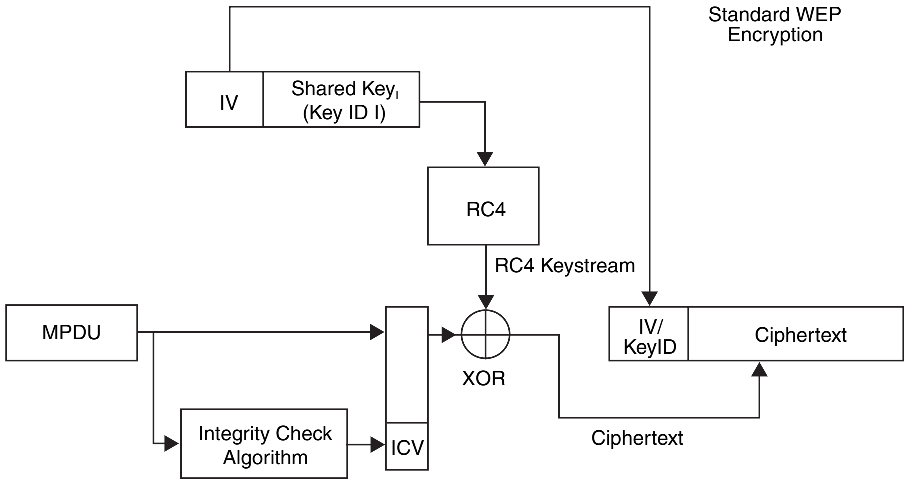


Figure 7.5: WEP

Step 2: Select a master key to be used from one of the four possible preshared secret keys. See Section 7.2.1 for the explanation of the four possible preshared secret keys.

Step 3: Select an IV and concatenate it with the master key to obtain a key seed. WEP does not specify how to select the IV. The IV selection process is left to the implementation.

Step 4: The key seed generated in Step 3 is then fed to an RC4 key-generator. The resulting RC4 key stream is then XORed with the MPDU + ICV generated in Step 1 to generate the ciphertext.

Step 5: A 4-byte header is then appended to the encrypted packet. It contains the 3-byte IV value and a 1-byte key-id specifying which one of the four preshared secret keys is being used as the master key.

The WEP process is now completed. An 802.11 header is then appended to this packet and it is ready for transmission. The format of this packet is shown in Figure 7.6.

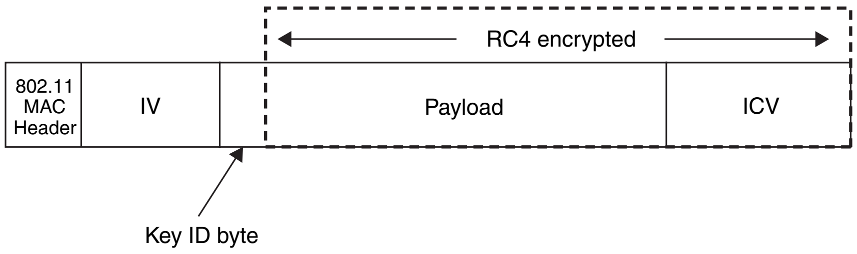


Figure 7.6: A WEP Packet

7.5.1 What's Wrong with WEP?

WEP uses RC4 (a stream cipher) in synchronous mode for encrypting data packets. Synchronous stream ciphers require that the key generators at the two communicating nodes must be kept synchronized by some external means because the loss of a single bit of a data stream encrypted under the cipher causes the loss of ALL data following the lost bit (for an explanation of this see Section 1.5). In brief, this is so because data loss desynchronizes the key stream generators at the two endpoints. Since data loss is widespread in the wireless medium, a synchronous stream cipher is not the right choice. This is one of the most fundamental problems of WEP. It uses a cipher not suitable for the environment it operates in.

It is important to re-emphasize here that the problem here is not the RC4 algorithm.[9] The problem is that a stream cipher is not suitable for a wireless medium where packet loss is widespread. SSL uses RC4 at the application layer successfully because SSL (and therefore RC4) operates over TCP (a reliable data channel) that does not lose any data packets and can therefore guarantee perfect synchronization between the two end points.

The WEP designers were aware of the problem of using RC4 in a wireless environment. They realized that due to the widespread data loss in the wireless medium, using a synchronous stream cipher across 802.11 frame boundaries was not a viable option. As a solution, WEP attempted to solve the synchronization problem of stream ciphers by shifting synchronization requirement from a session to a packet. In other words, since the synchronization between the end-points is not perfect (and subject to packet loss), 802.11 changes keys for every packet. This way each packet can be encrypted or decrypted irrespective of the previous packet's loss. Compare this with SSL's use of RC4, which can afford to use a single key for a complete TCP session. In effect, since the wireless medium is prone to data loss, WEP has to use a single packet as the synchronization unit rather than a complete session. This means that WEP uses a unique key for each packet.

Using a separate key for each packet solves the synchronization problem but introduces problems of its known. Recall that to create a per-packet key, the IV is simply concatenated with the master key. As a general rule in cryptography, the more exposure a key gets, the more it is susceptible to be compromised. Most security architectures therefore try to minimize the exposure of the master key when deriving secondary (session) keys from it. In WEP however, the derivation of the secondary (per-packet) key from the master key is too trivial (a simple concatenation) to hide the master key.

Another aspect of WEP security is that the IV which is concatenated with the master key to create the per-packet key is transmitted in cleartext with the packet too. Since the 24-bit IV is transmitted in the clear with each packet, an eavesdropper already has access to the first three bytes of the per-packet key.

The above two weaknesses make WEP susceptible to an Fluhrer-Mantin-Shamir (FMS) attack which uses the fact that simply concatenating the IV (available in plain text) to the master key leads to the generation of a class of RC4 weak keys. The FMS attack exploits the fact that the WEP creates the per-packet key by simply

[9] Though loopholes in the RC4 algorithm have been discovered too.

concatenating the IV with the master-key. Since the first 24 bits of each per-packet key is the IV (which is available in plain text to an eavesdropper),[10] the probability of using weak keys[11] is very high. Note that the FMS attack is a weakness in the RC4 algorithm itself. However, it is the way that the per-packet keys are constructed in WEP that makes the FMS attack a much more effective attack in 802.11 networks.

The FMS attack relies on the ability of the attacker to collect multiple 802.11 packets which have been encrypted with weak keys. Limited key space (leading to key re-use) and availability of IV in plaintext which forms the first 3 bytes of the key makes the FMS attack a very real threat in WEP. This attack is made even more potent in 802.11 networks by the fact that the first 8 bytes of the encrypted data in every packet are known to be the Sub-Network Access Protocol (SNAP) header. This means that simply XORing the first 2 bytes of the encrypted pay-load with the well known SNAP header yields the first 2 bytes of the generated key-stream. In the FMS attack, if the first 2 bytes of enough key-streams are known then the RC4 key can be recovered. Thus, WEP is an ideal candidate for an FMS attack.

The FMS attack is a very effective attack but is by no means the only attack which can exploit WEP weaknesses. Another such attack stems from the fact that one of the most important requirements of a synchronous stream cipher (like RC4) is that the same key should not be reused *EVER*. Why is it so important to avoid key reuse in RC4? Reusing the same key means that different packets use a common key stream to produce the respective ciphertext. Consider two packets of plaintext (P1 and P2) which use the same RC4 key stream for encryption.

Since $C1 = P1 \oplus RC4(key)$

And $C2 = P2 \oplus RC4(key)$

Therefore $C1 \oplus C2 = P1 \oplus P2$

Obtaining the XOR of the two plaintexts may not seem like an incentive for an attack but when used with frequency analysis techniques it is often enough to get lots of information about the two plaintexts. More importantly, as shown above, key reuse effectively leads to the effect of the key stream canceling out! An implication of this effect is that if one of the plaintexts (say P1) is known, P2 can be calculated easily since $P2 = (P1 \oplus P2) \oplus P1$. Another implication of this effect is that if an attacker

[10] Remember that each WEP packet carries the IV in plaintext format prepended to the encrypted packet.

[11] Use of certain key values leads to a situation where the first few bytes of the output are not all that random. Such keys are known as weak keys. The simplest example is a key value of 0.

(say, Eve) gets access to the <P1, C1> pair,[12] simply XORing the two produces the key stream K. Once Eve has access to K, she can decrypt C2 to obtain P2. Realize how the basis of this attack is the reuse of the key stream, K.

Now that we know why key reuse is prohibited in RC4, we look at what 802.11 needs to achieve this. Since we need a new key for every single packet to make the network really secure, 802.11 needs a very large key space, or rather a large number of unique keys. The number of unique keys available is a function of the key length. What is the key length used in WEP? Theoretically it is 64 bits. The devil, however, is in the details. How is the 64-bit key constructed? 24 bits come from the IV and 40 bits come from the base-key. Since the 40-bit master key never changes in most 802.11 deployments,[13] we must ensure that we use different IVs for each packet in order to avoid key reuse. Since the master key is fixed in length and the IV is only 24 bits long, the effective key length of WEP is 24 bits. Therefore, the key space for the RC4 is 2^N where N is the length of the IV. 802.11 specified the IV length as 24.

To put things in perspective, realize that if we have a 24 bit IV ($\rightarrow 2^{24}$ keys in the key-space), a busy base station which is sending 1500 byte-packets at 11 Mbps will exhaust all keys in the key space in $(1500*8)/(11*10^6*2^{24})$ seconds or about five hours. On the other hand, RC4 in SSL would use the same key space for 2^{24} ($= 10^7$) sessions. Even if the application has 10,000 sessions per day, the key space would last for three years. In other words, an 802.11 BS using RC4 has to reuse the same key in about five hours whereas an application using SSL RC4 can avoid key reuse for about three years. This shows clearly that the fault lies not in the cipher but in the way it is being used. Going beyond an example, analysis of WEP has shown that there is a 50% chance of key reuse after 4823 packets, and there is 99% chance of collision after 12,430 packets. These are dangerous numbers for a cryptographic algorithm.

Believe it or not, it gets worse. 802.11 specifies no rules for IV selection. This in turn means that changing the IV with each packet is optional. This effectively means that 802.11 implementations may use the same key to encrypt all packets without violating the 802.11 specifications. Most implementations, however, vary from randomly generating the IV on a per-packet basis to using a counter for IV generation. WEP does specify that the IV be changed "frequently." Since this is vague, it means that an implementation which generates per-packet keys (more precisely the per-MPDU key) is 802.11-compliant and so is an implementation which re-uses the same key across MPDUs.

[12] This is not as difficult as it sounds.
[13] This weakness stems from the lack of a key-establishment or key-distribution protocol in WEP.

7.6 Data Integrity in 802.11

To ensure that a packet has not been modified in transit, 802.11 uses an Integrity Check Value (ICV) field in the packet. ICV is another name for message integrity check (MIC). As we saw in Chapter 1, the idea behind the ICV/MIC is that the receiver should be able to detect data modifications or forgeries by calculating the ICV over the received data and comparing it with the ICV attached in the message. Figure 7.7 shows the complete picture of how WEP and CRC32 work together to create the MPDU for transmission.

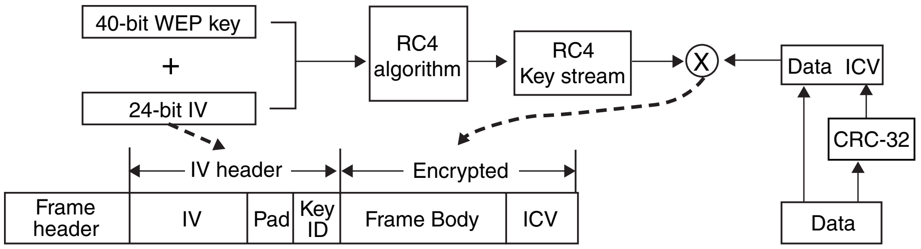


Figure 7.7: Data Integrity in WEP

The underlying assumption is that if Eve modifies the data in transit, she should not be able to modify the ICV appropriately to force the receiver into accepting the packet. In WEP, ICV is implemented as a Cyclic Redundancy Check-32 bits (CRC-32) checksum which breaks this assumption. The reason for this is that CRC-32 is linear and is not cryptographically computed, i.e., the calculation of the CRC-32 checksum does not use a key/shared secret. Also, this means that the CRC32 has the following interesting property:

$$CRC(X \oplus Y) = CRC(X) \oplus CRC(Y)$$

Now, if X represents the payload of the 802.11 packet over which the ICV is calculated, the ICV is CRC(X) which is appended to the packet. Consider an intruder who wishes to change the value of X to Z. To do this, they calculate $Y = X \oplus Z$. Then she captures the packet from the air-interface, XORs X with Y and the XORs the ICV with CRC(Y). Therefore, the packet changes from {X, CRC(X)} to $\{X \oplus Y, CRC(X) \oplus CRC(Y)\}$ or simply $\{X \oplus Y, CRC(X \oplus Y)\}$. If the intruder now re-transmits the packets to the receiver, the receiver would have no way of telling that the packet was modified in transit. This means that we can change bits in the payload of the packet while preserving the integrity of the packet if we also change the corresponding bits in the ICV of the packet.

Note that an attack like the one described above works because flipping bit x in the message results in a deterministic set of bits in the CRC that must be flipped to produce the correct checksum of the modified message. This property stems from the linearity of the CRC32 algorithm.

Realize that even though the ICV is encrypted (cryptographically protected) along with the rest of the payload in the packet, it is not cryptographically computed; that is, calculating the ICV does not involve keys and cryptographic operations. Simply encrypting the ICV does not prevent an attack like the one discussed above. This is so because the flipping of a bit in the ciphertext carries through after the RC4 decryption into the plaintext because $RC4(k, X \oplus Y) = RC4(k, X) \oplus Y$ and therefore:

$$RC4(k, CRC(X \oplus Y)) = RC4(k, CRC(X)) \oplus CRC(Y)$$

The problem with the message integrity mechanism specified in 802.11 is not only that it uses a linear integrity check algorithm (CRC32) but also the fact that the ICV does not protect all the information that needs to be protected from modification. Recall from Section 7.5 that the ICV is calculated over the MPDU data; in other words, the 802.11 header is not protected by the ICV. This opens the door to redirection attacks as explained below.

Consider an 802.11 BSS where an 802.11 STA (Alice) is communicating with a wired station (Bob). Since the wireless link between Alice and the access point (AP) is protected by WEP and the wired link between Bob and access point is not,[14] it is the responsibility of the AP to decrypt the WEP packets and forward them to Bob. Now, Eve captures the packets being sent from Alice to Bob over the wireless link. She then modifies the destination address to another node, say C (Charlie), in the 802.11 header and retransmits them to the AP. Since the AP does not know any better, it decrypts the packet and forwards it to Charlie. Eve, therefore, has the AP decrypt the packets and forward them to a destination address of choice.

The simplicity of this attack makes it extremely attractive. All Eve needs is a wired station connected to the AP and she can eavesdrop on the communication between Alice and Bob without needing to decrypt any packets herself. In effect, Eve uses the infrastructure itself to decrypt any packets sent from an 802.11 STA via an AP. Note that this attack does not necessarily require that one of the communicating stations be a wired station. Either Bob or Charlie (or both) could as easily be other 802.11 STAs which do not use WEP. The attack would still hold since the responsibility of

[14] WEP is an 802.11 standard used only on the wireless link.

decryption would still be with the AP. The bottom line is that the redirection attack is possible because the ICV is not calculated over the 802.11 header. There is an interesting security lesson here. A system can't have confidentiality without integrity, since an attacker can use the redirection attack and exploit the infrastructure to decrypt the encrypted traffic.

Another problem which stems from the weak integrity protection in WEP is the threat of a replay attack. A replay attack works by capturing 802.11 packets transmitted over the wireless interface and then replaying (retransmitting) the captured packet(s) later on with (or without) modification such that the receiving station has no way to tell that the packet it is receiving is an old (replayed) packet. To see how this attack can be exploited, consider a hypothetical scenario where Alice is an account holder, Bob is a bank and Eve is another account holder in the bank. Suppose Alice and Eve do some business and Alice needs to pay Eve $500. So, Alice connects to Bob over the network and transfers $500 from her account to Eve. Eve, however, is greedy. She knows Alice is going to transfer money. So, she captures all data going from Alice to Bob. Even though Eve does not know what the messages say, she has a pretty good guess that these messages instruct Bob to transfer $500 from Alice's account to Eve's. So, Eve waits a couple of days and replays these captured messages to Bob. This may have the effect of transferring another $500 from Alice's account to Eve's account unless Bob has some mechanism for determining that he is being replayed the messages from a previous session.

Replay attacks are usually prevented by linking the integrity protection mechanism to either timestamps and/or session sequence numbers. However, WEP does not provide for any such protection.

7.7 Loopholes in 802.11 Security

To summarize, here is the list of things that are wrong with 802.11 security:

1. 802.11 does not provide any mechanism for key establishment over an unsecure medium. This means key sharing among STAs in a BSS and sometimes across BSSs.
2. WEP uses a synchronous stream cipher over a medium, where it is difficult to ensure synchronization during a complete session.
3. To solve the previous problem, WEP uses a per-packet key by concatenating the IV directly to the preshared key to produce a key for RC4. This exposes the base key or master key to attacks like FMS.

4. Since the master key is usually manually configured and static and since the IV used in 802.11 is just 24 bits long, this results in a very limited key-space.
5. 802.11 specifies that changing the IV with each packet is optional, thus making key reuse highly probable.
6. The CRC-32 used for message integrity is linear.
7. The ICV does not protect the integrity of the 802.11 header, thus opening the door to redirection attacks.
8. There is no protection against replay attacks.
9. There is no support for a STA to authenticate the network.

Note that the limited size of the IV figures much lower in the list than one would expect. This emphasizes the fact that simply increasing the IV size would not improve WEP's security considerably. The deficiency of the WEP encapsulation design arises from attempts to adapt RC4 to an environment for which it is poorly suited.

7.8 WPA

When the loopholes in WEP, the original 802.11 security standard, had been exposed, IEEE formed a Task Group: 802.11i with the aim of improving upon the security of 802.11 networks. This group came up with the proposal of a Robust Security Network (RSN). A RSN is an 802.11 network which implements the security proposals specified by the 802.11i group and allows only RSN-capable devices to join the network, thus allowing no "holes." The term *hole* is used to refer to a non-802.11i compliant STA which by virtue of not following the 802.11i security standard could make the whole network susceptible to a variety of attacks.

Since making a transition from an existing 802.11 network to a RSN cannot always be a single-step process (we will see why in a moment), 802.11i allows for a Transitional Security Network (TSN) which allows for the existence of both RSN and WEP nodes in an 802.11 network. As the name suggests, this kind of a network is specified only as a transition point and all 802.11 networks are finally expected to move to a RSN. The terms RSN and 802.11i are sometimes used interchangeably to refer to this security specification.

The security proposal specified by the Task Group-i uses the Advanced Encryption Standard (AES) in its default mode. One obstacle in using AES is that it is not backward compatible with existing WEP hardware. This is so because AES requires the existence of a new more powerful hardware engine. This means that there is also a need for a security solution which can operate on existing hardware. This was a pressing need for vendors of 802.11 equipment. This is where the Wi-Fi alliance came into the picture.

The Wi-Fi alliance is an alliance of major 802.11 vendors formed with the aim of ensuring product interoperability. To improve the security of 802.11 networks without requiring a hardware upgrade, the Wi-Fi alliance adopted Temporal Key Integrity Protocol (TKIP) as the security standard that needs to be deployed for Wi-Fi certification. This form of security has therefore come to be known as Wi-Fi Protected Access (WPA). WPA is basically a prestandard subset of 802.11i which includes the key management and the authentication architecture (802.1X) specified in 802.11i. The biggest difference between WPA and 802l.11i (which has also come to be known as WPA2) is that instead of using AES for providing confidentiality and integrity, WPA uses TKIP and MICHAEL respectively. We look at TKIP/WPA in this section and the 802.11i/WPA2 using AES in the next section.

TKIP stands for Temporal Key Integrity Protocol. It was designed to fix WEP loopholes while operating within the constraints of existing 802.11 equipment (APs, WLAN cards and so on). To understand what we mean by the "constraints of existing 802.11 hardware," we need to dig a little deeper. Most 802.11 equipment consists of some sort of a WLAN Network Interface Card (NIC) (also known as WLAN adapter) which enables access to an 802.11 network. A WLAN NIC usually consists of a small microprocessor, some firmware, a small amount of memory and a special-purpose hardware engine. This hardware engine is dedicated to WEP implementation since software implementations of WEP are too slow. To be precise, the WEP encryption process is implemented in hardware. The hardware encryption takes the IV, the base (master) key and the plaintext data as the input and produces the encrypted output (ciphertext). One of the most severe constraints for TKIP designers was that the hardware engine cannot be changed. We see in this section how WEP loopholes were closed given these constraints.

7.8.1 Key Establishment

One of the biggest WEP loopholes is that it specifies no key-establishment protocol and relies on the concept of preshared secret keys which should be established using some out-of-band mechanism. Realize that this is a system architecture problem. In other words, solving this problem requires support from multiple components (the AP, the STA and usually also a backend authentication server) in the architecture.

One of the important realizations of the IEEE 802.11i task group was that 802.11 networks were being used in two distinct environments: the home network and the enterprise network. These two environments had distinct security requirements and different infrastructure capacities to provide security. Therefore, 802.11i specified two distinct security architectures. For the enterprise network, 802.11i specifies the use of IEEE 802.1X for key establishment and authentication. As we will see in our discussion in the next section, 802.1X requires the use of a backend authentication server. Deploying a back end authentication server is not usually feasible in a home environment. Therefore, for home deployments of 802.11, 802.11i allows the use of the "out-of-band mechanism" (read manual configuration) for key establishment.

We look at the 802.1X architecture in the next section and see how it results in the establishment of a Master Key (MK). In this section, we assume that the two communicating end-points (the STA and the AP) already share a MK which has either been configured manually at the two end-points (WEP architecture) or has been established using the authentication process (802.1X architecture). This section looks at how this MK is used in WPA.

Recall that a major loophole in WEP was the manner[15] in which this master key was used which made it vulnerable to compromise. WPA solves this problem by reducing the exposure of the master key, thus making it difficult for an attacker to discover the master key. To achieve this, WPA adds an additional layer to the key hierarchy used in WEP. Recall from Section 6.4 that WEP uses the master key for authentication and to calculate the per-packet key. In effect there is a two-tier key hierarchy in WEP: the master (preshared secret) key and the per-packet key.

WPA extends the two-tier key-hierarchy of WEP to a multitier hierarchy (See Figure 7.8). At the top level is still the master key, referred to as the Pair-wise Master Key (PMK) in WPA. The next level in the key hierarchy is the PTK which is derived from the PMK. The final level is the per-packet keys which are generated by feeding the PTK to a key-mixing function. Compared with the two-tier WEP key hierarchy, the three-tier key hierarchy of WPA avoids exposing the PMK in each packet by introducing the concept of PTK.

As we saw, WPA is flexible about how the master key (PMK in WPA) is established. The PMK, therefore, may be a preshared[16] secret key (WEP-design) or a key derived from an authentication process like 802.1X.[17] WPA does require that the PMK be

[15] The per-packet key is obtained by simply concatenating the IV with the preshared secret key. Therefore, a compromised per-packet key exposes the preshared secret key.

[16] As we saw, this usually means that the keys are manually configured.

[17] It is expected that most enterprise deployments of 802.11 would use 802.1X while the preshared secret key method (read manual configuration) would be used by residential users.

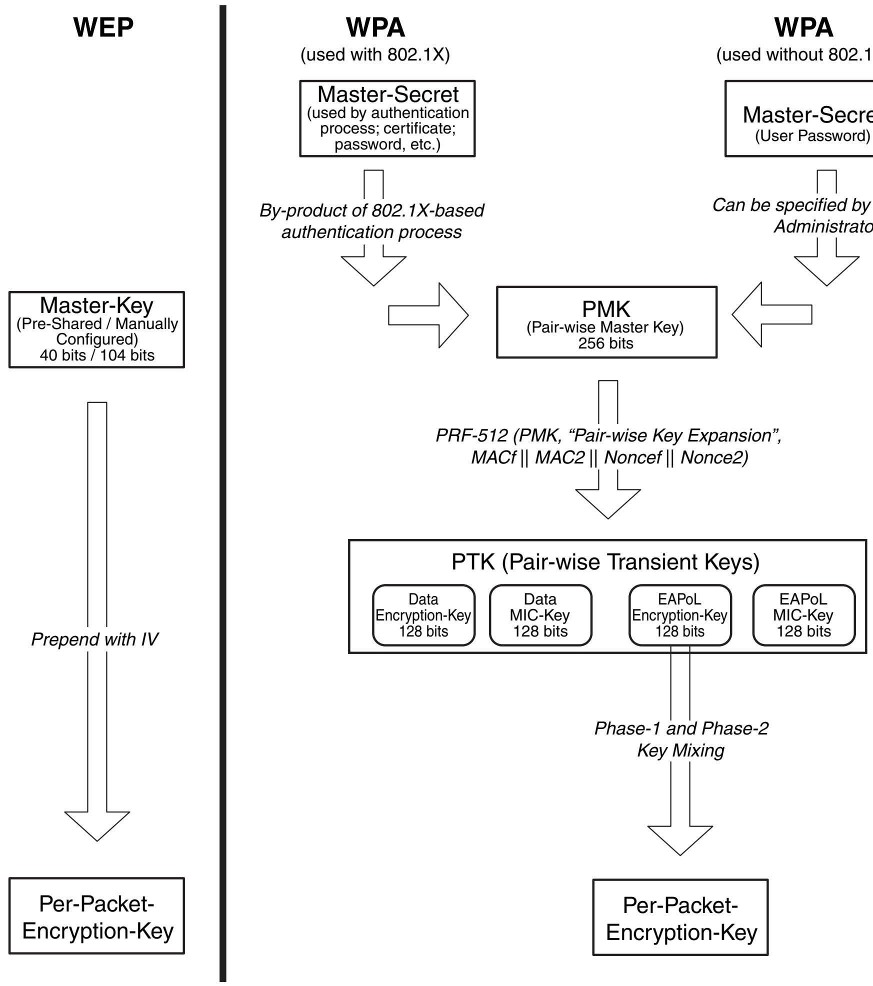

WEP
WPA
(used with 802.1X)
WPA
(used without 802.1X)
Master-Secret
(used by authentication process; certificate; password, etc.)
Master-Secret
(User Password)
By-product of 802.1X-based authentication process
Can be specified by Network Administrator
Master-Key
(Pre-Shared / Manually Configured)
40 bits / 104 bits
PMK
(Pair-wise Master Key)
256 bits
PRF-512 (PMK, "Pair-wise Key Expansion", MACf || MAC2 || Noncef || Nonce2)
PTK (Pair-wise Transient Keys)
Data Encryption-Key 128 bits
Data MIC-Key 128 bits
EAPoL Encryption-Key 128 bits
EAPoL MIC-Key 128 bits
Prepend with IV
Phase-1 and Phase-2 Key Mixing
Per-Packet-Encryption-Key
Per-Packet-Encryption-Key

Figure 7.8: Key Hierarchy in 802.11

256 bits (or 32 bytes) long. Since a 32-byte key is too long for humans to remember, 802.11 deployments using preshared keys may allow the user to enter a shorter password which may then be used as a seed to generate the 32-byte key.

The next level in the key hierarchy after the PMK are the PTK. WPA uses the PMK for deriving the Pair-wise Transient Keys (PTK) which are basically session keys. The term PTK is used to refer to a set of session keys which consists of four keys, each of which is 128 bits long. These four keys are as follows: an encryption key for data, an integrity key for data, an encryption key for EAPoL messages and an integration key for EAPoL messages. Note that the term *session* here refers to the association between a STA and an AP. Every time a STA associates with an AP, it is the beginning of a new session and this results in the generation of a new PTK (set of keys) from the PMK. Since the session keys are valid only for a certain period of time, they are also referred to as temporal keys and the set of four session keys together is referred to as the Pair-wise Transient Keys (PTK). The PTK are derived from the PMK using a Pseudorandom Function (PRF). The PRFs used for derivation of PTKs (and nonces) are explicitly specified by WPA and are based on the HMAC-SHA algorithm.

PTK = PRF-512(PMK, “Pair-wise key expansion”, AP_MAC || STA_MAC || ANonce || SNonce)

Realize that to obtain the PTK from the PMK we need five input values: the PMK, the MAC addresses of the two endpoints involved in the session and one nonce each from the two endpoints. The use of the MAC addresses in the derivation of the PTK ensures that the keys are bound to sessions between the two endpoints and increases the effective key space of the overall system.

Realize that since we want to generate a different set of session keys from the same PMK for each new session,[18] we need to add another input into the key generation mechanism which changes with each session. This input is the nonce. The concept of nonce is best understood by realizing that it is short for *Number-Once*. The value of nonce is thus arbitrary except that a nonce value is never used again.[19] Basically it is a number which is used only once. In our context, a nonce is a unique number (generated randomly) which can distinguish between two sessions established between a given STA and an AP at different points in time. The two nonces involved in PTK generation are generated, one each, by the two end points involved in the session; i.e., the STA (SNonce) and the AP (ANonce). WPA specifies that a nonce should be generated as follows:

ANonce = PRF-256(Random Number, “Init Counter”, AP_MAC || Time)
SNonce = PRF-256(Random Number, “Init Counter”, STA_MAC || Time)

The important thing to note is that the PTKs are effectively shared between the STA and the AP and are used by both the STA and the AP to protect the data/EAPoL-messages they transmit. It is therefore important that the input values required for derivation of PTK from the PMK come from *both* the STA and the AP. Note also that the key derivation process can be executed in parallel at both endpoints of the

[18] If a STA disconnects from the AP and connects back with an AP at a later time, these are considered two different sessions.
[19] To be completely accurate, nonce values are generated such that the probability of the same value being generated twice is very low.

session (the STA and the AP) once the Nonces and the MAC addresses have been exchanged. Thus, both the STA and the AP can derive the same PTK from the PMK simultaneously.

The next step in the key hierarchy tree is to derive per-packet keys from the PTK. WPA improves also upon this process significantly. Recall from Section 7.5 that the per-packet key was obtained by simply concatenating the IV with the master key in WEP. Instead of simply concatenating the IV with the master key, WPA uses the process shown in Figure 7.9 to obtain the per packet key. This process is known as per-packet key mixing and is shown in Figure 7.9.

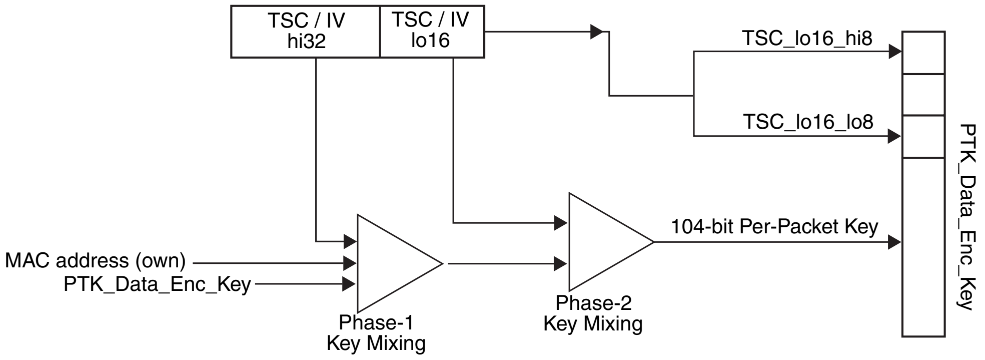


Figure 7.9: TKIP Encryption

In phase one, the session data encryption key is "combined" with the high order 32 bits of the IV and the MAC address. The output from this phase is "combined" with the lower order 16 bits of the IV and fed to phase two, which generates the 104-bit per-packet key. There are many important features to note in this process:

1. It assumes the use of a 48-bit IV (more of this in Section 7.8.2).
2. The size of the encryption key is still 104 bits, thus making it compatible with existing WEP hardware accelerators.
3. Since generating a per-packet key involves a hash operation which is computation intensive for the small MAC processor in existing WEP hardware, the process is split into two phases. The processing intensive part is done in phase one whereas phase two is much less computation intensive.
4. Since phase one involves the high order 32 bits of the IV, it needs to be done only when one of these bits change; that is, once in every 65,536 packets.
5. The key-mixing function makes it very hard for an eavesdropper to correlate the IV and the per-packet key used to encrypt the packet.

7.8.2 Authentication

As we said in the previous section, 802.11i specified two distinct security architectures. For the home network, 802.11i allows the manual configuration of keys just like WEP. For the enterprise network however, 802.11i specifies the use of IEEE 802.1X for key establishment and authentication. Sections 3.3.1 and 3.3.2 explain the 802.1X architecture in detail. We just summarize the 802.1X architecture in this section.

802.1X is closely architected along the lines of EAPoL (EAP over LAN). Figure 7.10a shows the conceptual architecture of EAPoL and Figure 7.10b shows the overall system architecture of EAPoL. The controlled port is open only when the device connected to the authenticator has been authorized by 802.1x. On the other hand, the uncontrolled port provides a path for extensible authentication protocol over LAN (EAPoL) traffic *ONLY*. Figure 7.10a shows how access to even the uncontrolled port may be limited using MAC filtering.[20] This scheme is sometimes used to deter DoS attacks.

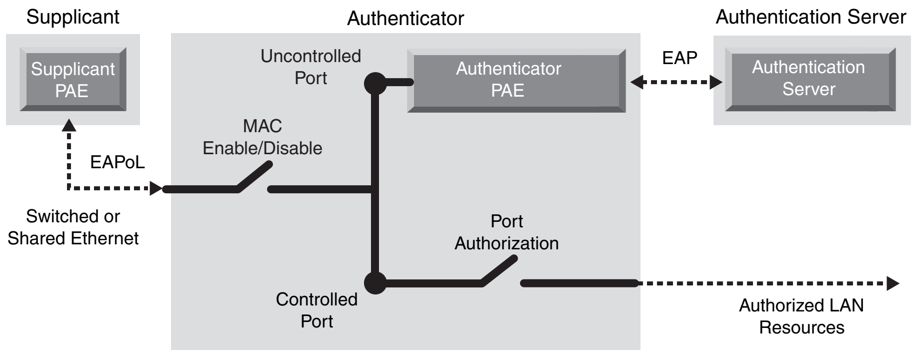


Figure 7.10a: 802.1X/EAP Port Model

EAP specifies three network elements: the supplicant, the authenticator and the authentication server. For EAPoverLAN, the end user is the supplicant, the Layer 2 (usually Ethernet) switch is the authenticator controlling access to the network using logical ports, and the access decisions are taken by the backend authentication server after carrying out the authentication process. Which authentication process to use (MD5, TLS and so on) is for the network administrator to decide.

EAPoL can be easily adapted to be used in the 802.11 environment as shown in Figure 7.10c. The STA is the supplicant, the AP is the authenticator controlling access to the network, and there is a backend authentication server. The analogy is all the more

[20] Allowing only STAs with have a MAC address which is "registered" or "known" to the network.

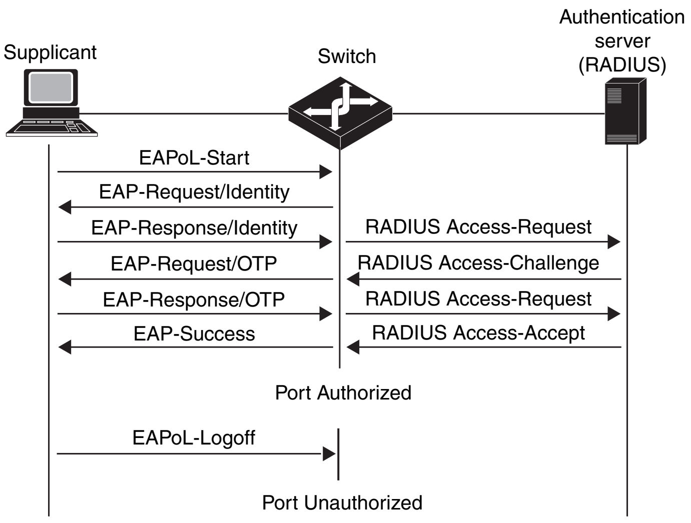


Figure 7.10b: EAPoL

striking if you consider that an AP is in fact just a Layer 2 switch, with a wireless and a wired interface.

There is however one interesting piece of detail that needs attention. The 802.1X architecture carries the authentication process between the supplicant (STA) and the backend authentication server.[21] This means that the master key (resulting from an authentication process like TLS) is established between the STA and backend server. However, confidentiality and integrity mechanisms in the 802.11 security architecture

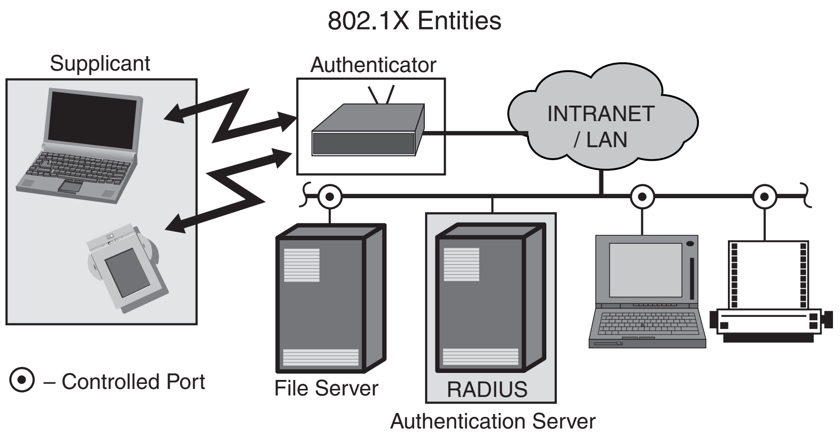


Figure 7.10c: EAP over WLAN

[21] With the AP controlling access to the network using logical ports.

are implemented between the AP and the STA. This means that the session (PTK) and per packet keys (which are derived from the PMK) are needed at the STA and the AP. The STA already has the PMK and can derive the PTK and the per-packet keys. However, the AP does not yet have the PMK. Therefore, what is needed is a mechanism to get the PMK from the authentication server to the AP securely.

Recall that in the 802.1X architecture, the result of the authentication process is conveyed by the authentication server to the AP so that the AP may allow or disallow the STA access to the network. The communication protocol between the AP and the authentication server is not specified by 802.11i but is specified by WPA to be RADIUS. Most deployments of 802.11 would probably end up using RADIUS. The RADIUS protocol does allow for distributing the key securely from the authentication server to the AP and this is how the PMK gets to the AP.

Note that 802.1X is a framework for authentication. It does not specify the authentication protocol to be used. Therefore, it is up to the network administrator to choose the authentication protocol they want to plug in to the 802.1X architecture. One of the most often discussed authentication protocols to be used with 802.1X is TLS. Section 3.3.3 discusses how the TLS protocol is used with EAPoL. Figure 7.10d

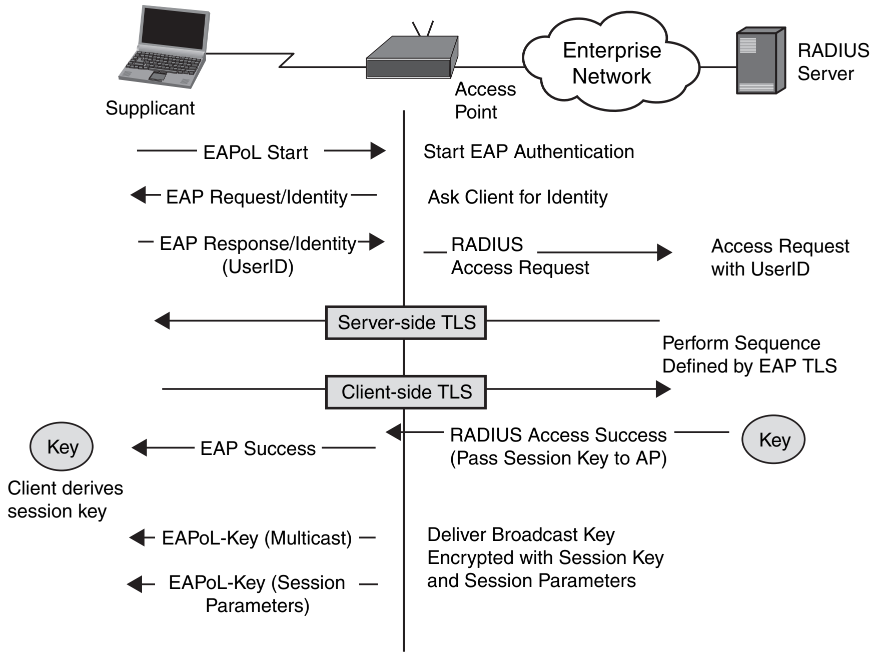

Supplicant
Access Point
Enterprise Network
RADIUS Server
EAPoL Start
Start EAP Authentication
EAP Request/Identity
Ask Client for Identity
EAP Response/Identity (UserID)
RADIUS Access Request
Access Request with UserID
Server-side TLS
Perform Sequence Defined by EAP TLS
Client-side TLS
Key
EAP Success
RADIUS Access Success (Pass Session Key to AP)
Key
Client derives session key
EAPoL-Key (Multicast)
EAPoL-Key (Session Parameters)
Deliver Broadcast Key Encrypted with Session Key and Session Parameters

Figure 7.10d: 802.1X Network Architecture

summarizes how TLS can be used as an authentication protocol in a EAP over WLAN environment. The EAP-TLS protocol is well documented. It has been analyzed extensively and no significant weaknesses have been found in the protocol itself. This makes it an attractive option for security use in 802.1X. However, there is a deployment issue with this scheme.

Note that EAP-TLS relies on certificates to authenticate the network to the clients and the clients to the networks. Requiring the network (the servers) to have certificates is a common theme in most security architectures. However, the requirement that each client be issued a certificate leads to the requirement of the wide spread deployment of PKI. Since this is sometimes not a cost effective option, a few alternative protocols have been proposed: EAP-TTLS (tunneled TLS) and PEAP. Both of these protocols use certificates to authenticate the network (the server) to the client but do not use certificates to authenticate the client to the server. This means that a client no longer needs a certificate to authenticate itself to the server: instead the clients can use password-based schemes (CHAP, PAP and so on) to authenticate themselves. Both protocols divide the authentication process in two phases. In phase 1, we authenticate the network (the server) to the client using a certificate and establish a TLS tunnel between the server and the client. This secure[22] TLS channel is then used to carry out a password-based authentication protocol to authenticate the client to the network (server).

7.8.3 Confidentiality

Recall from Section 7.5.1 that the fundamental WEP loophole stems from using a stream cipher in an environment susceptible to packet loss. To work around this problem, WEP designers changed the encryption key for each packet. To generate the per-packet encryption key, the IV was concatenated with the preshared key. Since the preshared key is fixed, it is the IV which is used to make each per-packet key unique. There were multiple problems with this approach.

First, the IV size at 24 bits was too short. At 24 bits there were only 16,777,216 values before a duplicate IV value was used. Second, WEP did not specify how to select an IV for each packet.[23] Third, WEP did not even make it mandatory to vary the IV on a per-packet basis—realize that this meant WEP explicitly allowed reuse of per-packet keys. Fourth, there was no mechanism to ensure that the IV was unique on a per station basis. This made the IV collision space shared between stations, thus making

[22] Secure since it protects the identity of the client during the authentication process.
[23] Implementations vary from a sequential increase starting from zero to generating a random IV for each packet.

a collision even more likely. Finally, simply concatenating the IV with the preshared key to obtain a per-packet key is cryptographically unsecure, making WEP vulnerable to the FMS attack. The FMS attack exploits the fact that the WEP creates the per-packet key by simply concatenating the IV with the master-key. Since the first 24 bits of each per-packet key is the IV (which is available in plain text to an eavesdropper),[24] the probability of using weak keys[25] is very high.

First off, TKIP doubles the IV size from 24 bits to 48 bits. This results in increasing the time to key collision from a few hours to a few hundred years. Actually, the IV is increased from 24 bits to 56 bits by requiring the insertion of 32 bits between the existing WEP IV and the start of the encrypted data in the WEP packet format. However, only 48 bits of the IV are used since eight bits are reserved for discarding some known (and some yet to be discovered) weak keys.

Simply increasing the IV length will, however, not work with the existing WEP hardware accelerators. Remember that existing WEP hardware accelerators expect a 24-bit IV as an input to concatenate with a preshared key (40/104-bit) in order to generate the per-packet key (64/128-bit). This hardware cannot be upgraded to deal with a 48-bit IV and generate an 88/156-bit key. The approach, therefore, is to use per-packet key mixing as explained in Section 7.8.1. Using the per-packet key mixing function (much more complicated) instead of simply concatenating the IV to the master key to generate the per-packet key increases the effective IV size (and hence improves on WEP security) while still being compatible with existing WEP hardware.

7.8.4 Integrity

WEP used CRC-32 as an integrity check. The problem with this protocol was that it was linear. As we saw in Section 7.6, this is not a cryptographically secure integrity protocol. It does however have the merit that it is not computation intensive. What TKIP aims to do is to specify an integrity protocol which is cryptographically secure and yet not computation intensive so that it can be used on existing WEP hardware which has very little computation power. The problem is that most well known protocols used for calculating a message integrity check (MIC) have lots of multiplication operations and multiplication operations are computation intensive. Therefore, TKIP uses a new MIC protocol—MICHAEL—which uses no multiplication operations and

[24] Remember that each WEP packet carries the IV in plain text format prepended to the encrypted packet.

[25] Use of certain key values leads to a situation where the first few bytes of the output are not all that random. Such keys are known as weak keys. The simplest example is a key value of 0.

relies instead on shift and add operations. Since these operations require much less computation, they can be implemented on existing 802.11 hardware equipment without affecting performance.

Note that the MIC value is added to the MPDU in addition to the ICV which results from the CRC32. It is also important to realize that MICHAEL is a compromise. It does well to improve upon the linear CRC-32 integrity protocol proposed in WEP while still operating within the constraints of the limited computation power. However, it is in no way as cryptographically secure as the other standardized MIC protocols like MD5 or SHA-1. The TKIP designers knew this and hence built in countermeasures to handle cases where MICHAEL might be compromised. If a TKIP implementation detects two failed forgeries (two packets where the calculated MIC does not match the attached MIC) in one second, the STA assumes that it is under attack and as a countermeasure deletes its keys, disassociates, waits for a minute and then re-associates. Even though this may sound a little harsh, since it disrupts communication, it does avoid forgery attacks.

Another enhancement that TKIP makes in IV selection and use is to use the IV as a sequence counter. Recall that WEP did not specify how to generate a per-packet IV.[26] TKIP explicitly requires that each STA start using an IV with a value of 0 and increment the value by one for each packet that it transmits during its session[27] lifetime. This is the reason the IV can also be used as a TKIP Sequence Counter (TSC). The advantage of using the IV as a TSC is to avoid the replay attack to which WEP was susceptible.

TKIP achieves replay protection by using a unique IV with each packet that it transmits during a session. This means that in a session, each new packet coming from a certain MAC address would have a unique number.[28] If each packet from Alice had a unique number, Bob could tell when Eve was replaying old messages. WEP does not have replay protection since it cannot use the IV as a counter. Why? Because WEP does not specify how to change IV from one packet to another and as we saw earlier, it does not even specify that you need to.

[26] In fact, WEP did not even specify that the IV had to be changed on a per-packet basis.
[27] An 802.11 session refers to the association between a STA and an AP.
[28] At least for 900 years—that's when the IV rolls over.

7.8.5 The Overall Picture: Confidentiality + Integrity

The overall picture of providing confidentiality and message integrity in TKIP is shown in Figure 7.10e.

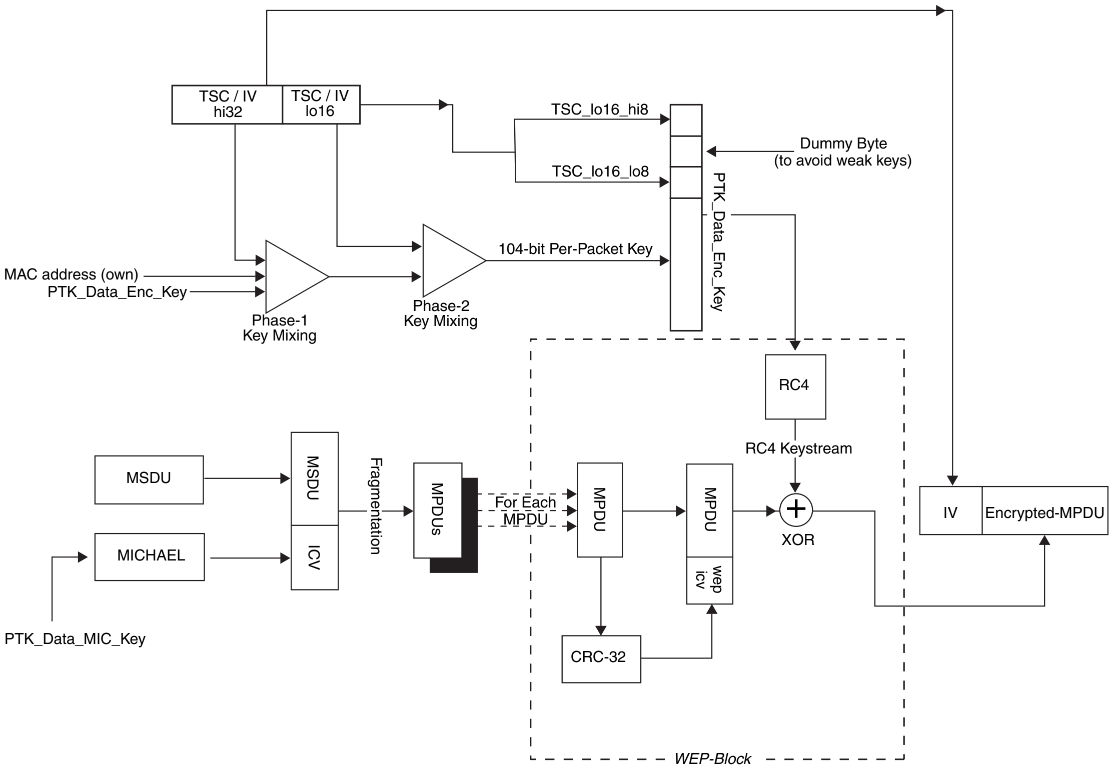


Figure 7.10e: TKIP—The Complete Picture

7.8.6 How Does WPA Fix WEP Loopholes?

In Section 7.7 we summarized the loopholes of WEP. At the beginning of Section 7.8 we said that WPA/TKIP was designed to close these loopholes while still being able to work with existing WEP hardware. In this section, we summarize what WPA/TKIP achieves and how.

WEP	WPA
Relies on preshared (out-of-band) key establishment mechanisms. Usually leads to manual configuration of keys and to key sharing among STAs in a BSS (often ESS).	Recommends 802.1X for authentication and key-establishment in enterprise deployments. Also supports preshared key establishment like WEP.
Uses a synchronous stream cipher which is unsuitable for the wireless medium.	Same as WEP.

WEP	WPA
Generates per-packet key by concatenating the IV directly to the master / preshared key thus exposing the base-key / master-key to attacks like FMS.	Solves this problem by (a) introducing the concept of PTK in the key hierarchy and (b) by using a key mixing function instead of simple concatenation to generate per-packet keys. This reduces the exposure of the master key.
Static master key + Small size of IV + Method of per-packet key generation → Extremely limited key space.	Increases the IV size to 56 bits and uses only 48 of these bits reserving 8-bits to discard weak keys. Also, use of PTK which are generated afresh for each new session increases the effective key space.
Changing the IV with each packet is optional → key-reuse highly probable.	Explicitly specifies that both the transmitter and the receiver initialize the IV to zero whenever a new set of PTK is established[29] and then increment it by one for each packet it sends.
Linear algorithm (CRC-32) used for message integrity → Weak integrity protection.	Replaces the integrity check algorithm to use MICHAEL which is nonlinear. Also, specifies countermeasures for the case where MICHAEL may be violated.
ICV does not protect the integrity of the 802.11 header → Susceptible to Redirection Attacks.	Extends the ICV computation to include the MAC source and destination address to protect against Redirection attacks.
No protection against replay attacks.	The use of IV as a sequence number provides replay protection.
No support for a STA to authenticate the network.	Use of 802.1X in enterprise deployments allows for this.

7.9 WPA2 (802.11i)

Recall from Section 7.8 that Wi-Fi protected access (WPA) was specified by the Wi-Fi alliance with the primary aim of enhancing the security of existing 802.11 networks by designing a solution which could be deployed with a simple software (firmware) upgrade and without the need for a hardware upgrade. In other words, WPA was a stepping stone to the final solution which was being designed by the IEEE 802.11i task group. This security proposal was referred to as the Robust Security Network (RSN) and also came to be known as the 802.11i security solution. The Wi-Fi alliance integrated this solution in their proposal and called it WPA2. We look at this security proposal in this section.

7.9.1 Key Establishment

WPA was a prestandard subset of IEEE 802.11i. It adopted the key-establishment, key hierarchy and authentication recommendations of 802.11i almost completely. Since WPA2 and 802.11i standard are the same, the key-establishment process and the key

[29] This usually happens every time the STA associates with an AP.

hierarchy architecture in WPA and WPA2 are almost identical. There is one significant difference though. In WPA2, the same key can be used for the encryption and integrity protection of data. Therefore, there is one less key needed in WPA2. For a detailed explanation of how the key hierarchy is established see Section 7.8.1.

7.9.2 Authentication

Just like key establishment and key hierarchy, WPA had also adopted the authentication architecture specified in 802.11i completely. Therefore, the authentication architecture in WPA and WPA2 is identical. For a detailed explanation of how the authentication architecture see Section 7.8.2.

7.9.3 Confidentiality

In this section we look at the confidentiality mechanism of WPA2 (802.11i). Recall that the encryption algorithm used in WEP was RC4, a stream cipher. Some of the primary weaknesses in WEP stemmed from using a stream cipher in an environment where it was difficult to provide lossless synchronous transmission. It was for this reason that Task Group i specified the use of a block encryption algorithm when redesigning 802.11 security. Since AES was (and still is) considered the most secure block cipher, it was an obvious choice. This was a major security enhancement since the encryption algorithm lies at the heart of providing confidentiality.

Recall from Section 1.4.4 that specifying an encryption algorithm is not enough for providing system security. What is also needed is to specify a mode of operation. To provide confidentiality in 802.11i, AES is used in the counter mode. Counter mode actually uses a block cipher as a stream cipher, thus combining the security of a block cipher with the ease of use of a stream cipher. Figure 7.11 shows how AES counter mode works.

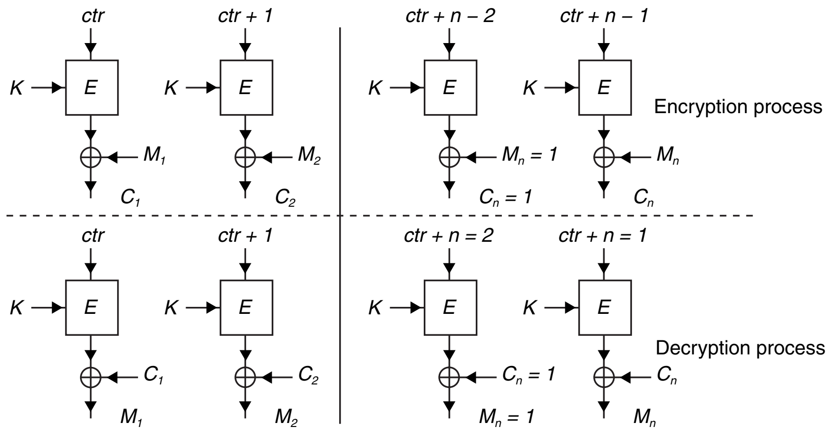


Figure 7.11: AES Counter Mode

Using the counter mode requires a counter. The counter starts at an arbitrary but predetermined value and is incremented in a specified fashion. The simplest counter operation, for example, would start the counter with an initial value of 1 and increment it sequentially by 1 for each block. Most implementations however, derive the initial value of the counter from a nonce value that changes for each successive message. The AES cipher is then used to encrypt the counter to produce a key stream. When the original message arrives, it is broken up into 128-bit blocks and each block is XORed with the corresponding 128 bits of the generated key stream to produce the ciphertext.

Mathematically, the encryption process can be represented as $C_i = M_i$ (+) E_k(i) where *i* is the counter. The security of the system lies in the counter. As long as the counter value is never repeated with the same key, the system is secure. In WPA2, this is achieved by using a fresh key for every session (See Section 7.8.1.).

To summarize, the salient features of AES in counter mode are as follows:

1. It allows a block cipher to be operated as a stream cipher.
2. The use of counter mode makes the generated key stream independent of the message, thus allowing the key stream to be generated before the message arrives.
3. Since the protocol by itself does not create any interdependency between the encryption of the various blocks in a message, the various blocks of the message can be encrypted in parallel if the hardware has a bank of AES encryption engines.
4. Since the decryption process is exactly the same as encryption,[30] each device only needs to implement the AES encryption block.
5. Since the counter mode does not require that the message be broken up into an exact number of blocks, the length of the encrypted text can be exactly the same as the length of the plain text message.

Note that the AES counter mode provides only for the confidentiality of the message and not the message integrity. We see how AES is used for providing the message integrity in the next section. Also, since the encryption and integrity protection processes are very closely tied together in WPA2/802.11i, we look at the overall picture after we have discussed the integrity process.

[30] XORing the same value twice leads back to the original value.

7.9.4 Integrity

To achieve message integrity, Task Group i extended the counter mode to include a Cipher Block Chaining (CBC)-MAC operation. This is what explains the name of the protocol: AES-CCMP where CCMP stands for Counter-mode CBC-MAC protocol. The CBC-MAC protocol (also known as CBC-residue) was first mentioned in Section 2.5.1. It is reproduced here in Figure 7.12 where the black boxes represent the encryption protocol (AES in our case).

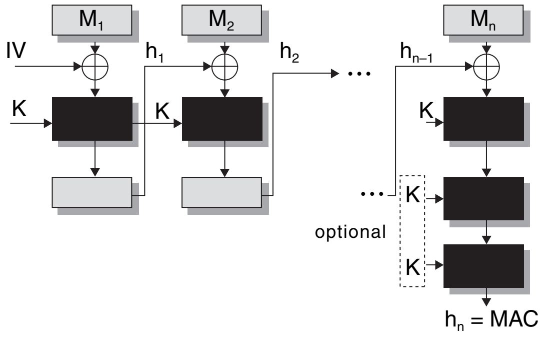


Figure 7.12: AES CBC-MAC

As shown in the figure, CBC-MAC XORs a plaintext block with the previous cipher block before encrypting it. This ensures that any change made to any cipher text (for example by a malicious intruder) block changes the decrypted output of the last block and hence changes the residue. CBC-MAC is an established technique for message integrity. What Task Group i did was to combine the counter mode of operation with the CBC-MAC integrity protocol to create the CCMP.

7.9.5 The Overall Picture: Confidentiality + Integrity

Since a single process is used to achieve integrity and confidentiality, the same key can be used for the encryption and integrity protection of data. It is for this reason that there is one less key needed in WPA2. The complete process which combines the counter mode encryption and CBC-MAC integrity works as follows.

In WPA2, the PTK is 384 bits long. Of this, the most significant 256 bits form the EAPoL MIC key and EAPoL encryption key. The least significant 128 bits form the data key. This data key is used for both encryption and integrity protection of the data. Before the integrity protection or the encryption process starts, a CCMP header is added to the 802.11 packet before transmission. The CCMP header is eight bytes in size. Of these eight bytes, six bytes are used for carrying the Packet Number (PN) which is needed for the other (remote) end to decrypt the packet and to verify the integrity of the packet. One byte is reserved for future use and the remaining byte contains the key ID. Note that the CCMP header is prepended to the payload of the packet and is not encrypted since the remote end needs to know the PN before it starts the decryption or the verification process. The PN is a per-packet sequence number which is incremented for each packet processed

The integrity protection starts with the generation of an Initialization Vector (IV) for the CBC-MAC process. This IV is created by the concatenation of the following entities: flag, priority, source MAC address, a PN and DLen as shown in Figure 7.13.

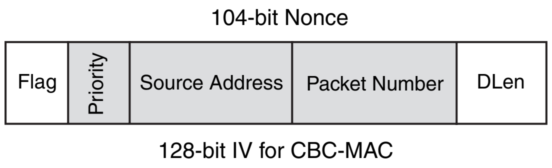


Figure 7.13: IV for AES CBC-MAC

The flag field has a fixed value of 01011001. The priority field is reserved for future use. The source MAC address is self explanatory and the packet number (PN) is as we discussed above. Finally, the last entity DLen indicates the data length of the plaintext. Note that the total length of the IV is 128 bits and the priority, source address and the packet number fields together also form the 104-bit nonce (shaded portion of Figure 7.13) which is required in the encryption process. The 128-bit IV forms the first block which is needed to start the CBC-MAC process described in Section 7.9.4. The CBC-MAC computation is done over the 802.11 header and the MPDU payload. This means that this integrity protection scheme also protects the source and the destination MAC address, the quality of service (QoS) traffic class and the data length. Integrity protecting the header along with the MPDU payload protects against replay attacks. Note that the CBC-MAC process requires an exact number of blocks to operate on. If the length of the plaintext data cannot be divided into an exact number of blocks, the plaintext data needs to be padded for the purposes of MIC computation.

Once the MAC has been calculated and appended to the MPDU, it is now ready for encryption. It is important to re-emphasize that only the data-part and the MAC part of the packet are encrypted whereas the 802.11 header and the CCMP header are not encrypted. From Section 7.9.3, we know that the AES-counter mode encryption process requires a key and a counter. The key is derived from the PTK as we discussed.

The counter is created by the concatenation of the following entities: Flag, Priority, Source MAC address, a packet number (PN) and Ctr as shown in Figure 7.14.

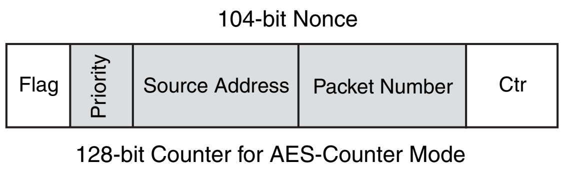


Figure 7.14: Counter for AES Counter Mode

Comparing Figure 7.14 with Figure 7.13, we see that the IV for the integrity process and the counter for the encryption process are identical except for the last sixteen bits. Whereas the IV has the last sixteen bits as the length of the plaintext, the counter has the last sixteen bits as Ctr. It is this Ctr which makes the counter a real "counter." The value of Ctr starts at one and counts up as the counter mode proceeds. Since the Ctr value is sixteen bits, this allows for up to 2^{16} (65,536) blocks of data in a MPDU. Given that AES uses 128-bit blocks, this means that an MPDU can be as long as 2^{23}, which is much more than what 802.11 allows, so the encryption process does not impose any additional restrictions on the length of the MPDU.

Even though CCMP succeeds in combining the encryption and integrity protocol in one process, it does so at some cost. First, the encryption of the various message blocks can no longer be carried out in parallel since CBC-MAC requires the output of the previous block to calculate the MAC for the current block. This slows down the protocol. Second, CBC-MAC requires the message to be broken into an exact number of blocks. This means that if the message cannot be broken into an exact number of blocks, we need to add padding bytes to it to do so. The padding technique has raised some security concerns among some cryptographers but no concrete deficiencies/attacks have been found against this protocol.

The details of the overall CCMP are shown in Figure 7.15 and finally the following table compares the WEP, WPA and WPA2 security architectures:

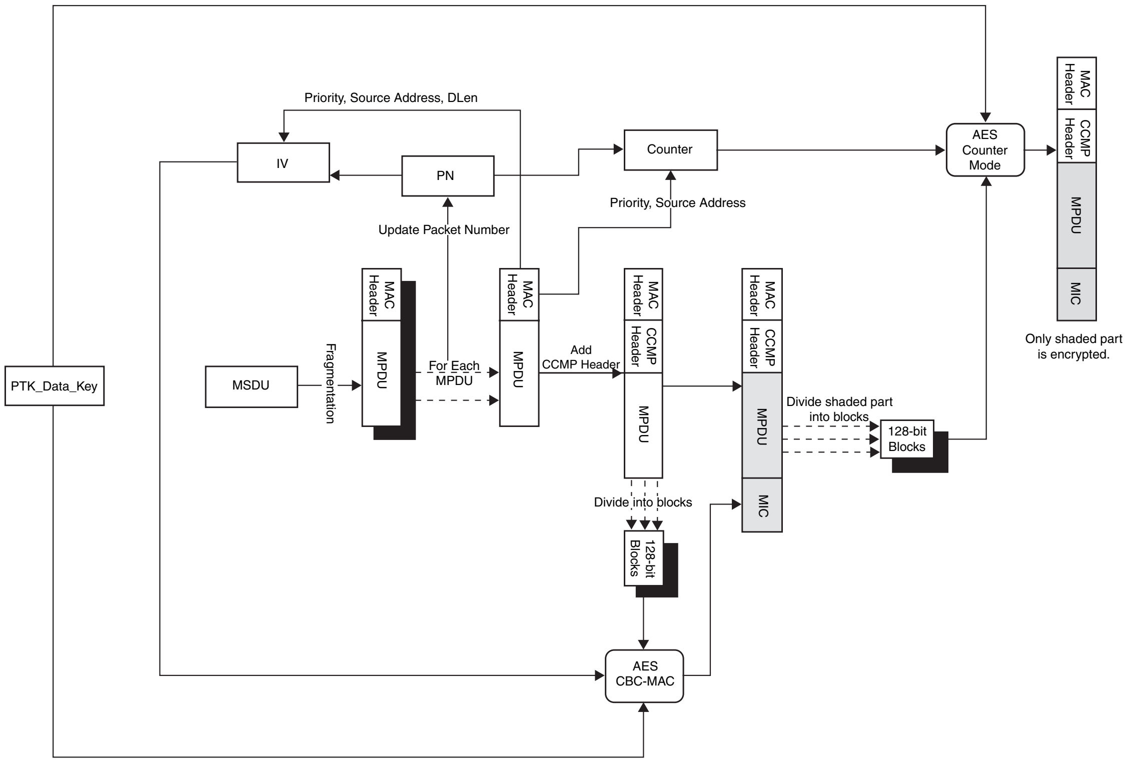


Figure 7.15: WPA2—The Complete Picture

WEP	WPA	WPA2
Relies on preshared a.k.a. out-of-band key establishment mechanisms. Usually leads to manual configuration of keys and to key sharing among STAs in a BSS (often ESS).	Recommends 802.1X for authentication and key-establishment in enterprise deployments. Also supports preshared key establishment like WEP.	Same as WPA.
Uses a synchronous stream cipher which is unsuitable for the wireless medium.	Same as WEP.	Replaces a stream cipher (RC4) with a strong block cipher (AES).
Generates per-packet key by concatenating the IV directly to the master/preshared key thus exposing the base-key/master-key to attacks like FMS.	Solves this problem (a) by introducing the concept of PTK in the key hierarchy and (b) by using a key mixing function instead of simple concatenation to generate per-packet keys. This reduces the exposure of the master key.	Same as WPA.
Static master key + Small size of IV + Method of per-packet key generation → Extremely limited key space.	Increases the IV size to 56 bits and uses only 48 of these bits reserving 8-bits to discard weak keys. Also, use of PTK which are generated afresh for each new session increases the effective key space.	Same as WPA.

WEP	WPA	WPA2
Changing the IV with each packet is optional → key-reuse highly probable.	Explicitly specifies that both the transmitter and the receiver initialize the IV to zero whenever a new set of PTK is established[31] and then increment it by one for each packet it sends.	Same as WPA.
Linear algorithm (CRC-32) used for message integrity → Weak integrity protection.	Replaces the integrity check algorithm to use MICHAEL which is nonlinear. Also, specifies countermeasures for the case where MICHAEL may be violated.	Provides for stronger integrity protection using AES-based CCMP.
ICV does not protect the integrity of the 802.11 header → Susceptible to Redirection Attacks.	Extends the ICV computation to include the MAC source and destination address to protect against Redirection attacks.	Same as WPA.
No protection against replay attacks.	The use of IV as a sequence number provides replay protection.	Same as WPA.
No support for a STA to authenticate the network.	No explicit attempt to solve this problem but the recommended use of 802.1X could be used by the STA to authenticate the network.	Same as WPA.

[31] This usually happens every time the STA associates with an AP.

CHAPTER 8

Security in Wireless Ad Hoc Networks

8.1 Introduction

The term *Ad Hoc Networks* refers to networks which are formed on-the-fly (ad hoc), in other words on an as-needed basis.

The term *Ad Hoc Wireless Networks* refers to those ad hoc networks which use a wireless medium for communication. Since a wired ad hoc network would be synonymous with a LAN, the term *ad hoc networks* almost always means ad hoc wireless networks and the two terms are used interchangeably throughout this text.

The term *Mobile Ad Hoc NETworks (MANETs)* refers to ad hoc networks in which the nodes forming the ad hoc network are mobile. Most ad hoc networks allow their nodes to be mobile and are therefore MANETs.

In other words, these networks are formed on an as-needed basis and do not require the existence of any infrastructure. This property makes ad hoc wireless networks suitable for use in various scenarios like disaster recovery, enemy battlefields or in areas where user density is too sparse or too rare to justify the deployment of network infrastructure economically. Figure 8.1 shows some examples of ad hoc wireless networks.

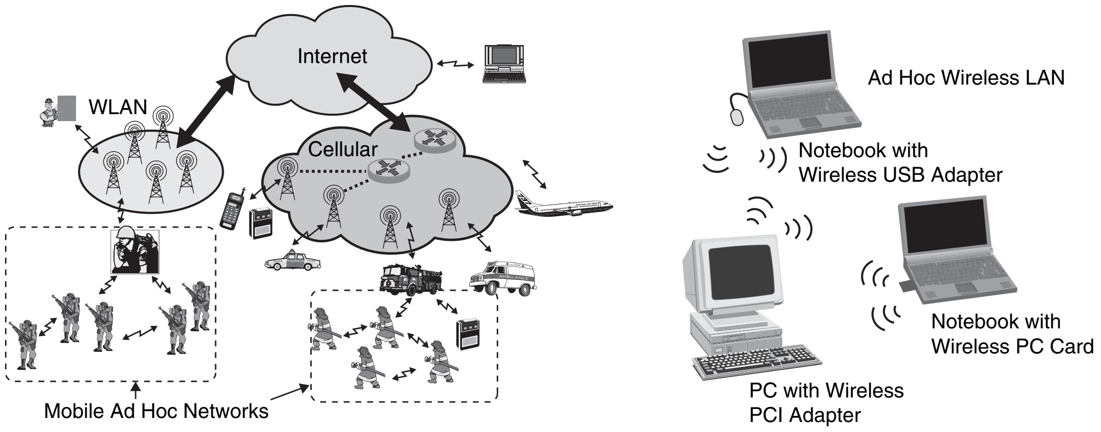


Figure 8.1: Examples of Ad Hoc Networks

The scenarios and examples shown in Figure 8.1 present a small subset of scenarios where ad hoc networks may be useful. An ad hoc network may operate in a stand-alone fashion or may be connected to a larger network like the Internet. Since ad hoc networks have such varied areas of use, it is instructive to classify them based on certain features.

First, ad hoc networks may be classified on the basis of their geographical coverage. Therefore we have ad hoc personal area networks (PANs), ad hoc local area networks (LANs) and ad hoc wide area networks (WANs).

Second, ad hoc networks may be classified based on whether or not nodes in the network are capable of acting as routers. To understand this classification, realize that the wireless networks that we have looked at till now in this book always used the fixed, static, wired infrastructure for routing. In TWNs, call routing was achieved by dedicated routing switches of the PSTN and the core GSM network (which consisted of MSCs and GMSCs). Furthermore, since both the PSTN and the core GSM network are wired networks which are static (that is, their network topology almost never changes), it is relatively easy to proactively distribute the network topology information to the routing switches. This in turn allows each routing switch to precompute and maintain routes to other switches, thus facilitating routing. Similarly, in wireless local area networks (WLANs), packet routing is achieved by using Layer 2 switches and IP routers. Again, since these routing devices are connected by a wired infrastructure and are static, it is relatively easy to proactively distribute the network topology information to the routers and switches.

Ad hoc networks have two major limitations: a) there are no dedicated routing devices (since there is no infrastructure available) and b) the network topology may change rapidly and unpredictably as nodes move. In the absence of any routing infrastructure, the nodes forming the ad hoc networks themselves have to act as routers. A MANET may therefore be defined as an autonomous system of mobile routers (and associated hosts) connected by wireless links—the union of which forms an arbitrary graph. Given the central importance of routing in ad hoc networks, it is not surprising that routing forms a basis for classifying ad hoc networks into two groups: single-hop ad hoc networks and multihop ad hoc networks. Single-hop ad hoc networks are ad hoc networks where nodes do not act as routers and therefore communication is possible only between nodes which are within each other's Radio Frequency (RF) range. On the other hand, multihop ad hoc networks are ad hoc networks where nodes are willing to act as routers and route or forward the traffic of other nodes.

If you look at the basis for two classifications closely; that is, the geographical coverage and the routing capability of nodes, the two classifications are not completely orthogonal. Ad hoc PANs are more likely to be single hop ad hoc networks since nodes would be close enough to be within each other's RF range. On the other hand, ad hoc LANs and ad hoc WANs are more likely to require nodes to have routing capability and therefore form multihop networks.

Multihop ad hoc networks and their security is an active area of research as of the writing of this book. In Sections 8.2–8.5 we look at some of the emerging security concepts (and areas of active research) in multihop ad hoc networks. Single-hop ad hoc networks are now being used commercially and one of the most popular single-hop ad hoc wireless standard is Bluetooth. We look at Bluetooth and how it implements security in Section 8.4.

8.2 Routing in Multihop Ad Hoc Networks

As we saw in the last section, routing is a huge challenge for multihop ad hoc networks. Not surprisingly, ensuring secure routing is an even bigger challenge. We start by looking at some of the most important routing protocols for multihop ad hoc networks in Section 8.2.1. Section 8.2.2 looks at some attacks which can be launched on routing in ad hoc networks and finally Section 8.2.3 looks at some possible solutions to these attacks.

8.2.1 Proactive Routing

Proactive routing protocols modify existing link-state or distance-vector-based routing protocols used in static infrastructure routing today to include support for mobility. Most protocols in this category use periodic messages to distribute information about the current network topology. Each router then uses this network topology information to compute and maintain routes to various destinations in the network. The aim of these protocols is for each router to have a valid route[1] to each destination at all times. When used in multihop ad hoc networks where each node is a router, this means that each node aims to know the route to other nodes in the network.

The advantage of using these protocols is that when a node needs to send a packet, it can refer to its routing table to find out how to get the packet there. This leads to extremely short transmission delays (only as long as it takes to look up the route from the table). On the other hand, the primary disadvantage of these protocols is that the periodic updates that are needed for distributing network topology information cost

[1] This may either be the complete route to the destination or just the next hop to the destination.

bandwidth and battery life (since nodes have to wake up periodically to broadcast and process the periodic messages).

Considering the advantages and disadvantages of proactive routing protocols, we can conclude that these protocols are most suitable in ad hoc networks where the number of nodes is small and nodes have limited mobility.

8.2.2 Reactive Routing

Unlike proactive routing protocols, where each node aims to maintain a route to all other nodes at all times, reactive routing protocols work by computing a route only when it is needed. So, when a node has a packet to transmit, it first discovers the route to the destination (usually by broadcasting messages) and then sends out the message.

Since these protocols do not require periodic transmission of messages, one of the primary advantages of using reactive routing protocols is the savings in bandwidth and battery life as compared to proactive routing protocols. On the other hand, the primary disadvantage of these protocols is that packets have to wait while the node tries to find the route to the destination thus leading to long transmission delays.

Considering the advantages and disadvantages of reactive routing protocols, we can conclude that these protocols are most suitable for ad hoc networks where the network topology is dynamic and/or where there are a large number of nodes in the network.

8.2.3 Hybrid Routing

Hybrid routing protocols attempt to combine the advantages of proactive and reactive protocols. A popular example is the Zone Routing Protocol (ZRP) which works by dividing the network into zones. Within a zone, reactive routing protocols are used to cope with frequent node mobility. Inter-zone messages must be routed via the zone gateway. The zone gateways of all zones form a tier-2 network and use proactive routing among themselves since this tier-2 network is assumed to be relatively static. In summary, tier-1 (zones) uses reactive routing and tier-2 uses proactive routing.

8.2.4. Routing Attacks

There are quite a few routing protocols for multihop ad hoc data networks today that cope well with the dynamically changing topology of ad hoc networks. However, irrespective of which protocol is being used for routing, this routing in ad hoc networks is based on cooperation among nodes in the network. This cooperation assumes an inherent trust relationship among nodes—which is never a good security approach, as we know. It is this inherent trusting of other nodes for routing that makes routing an attractive target for attacks.

Attacks on routing protocols can come from either nodes which are not part of the network (external attacks) or from nodes which are part of the network but which have been compromised (internal attacks). Both external and internal attackers may launch routing attacks by injecting erroneous routing information, replaying old (outdated) routing information or by distorting routing information. Such attacks may lead to unintended network partitioning, excessive traffic load, loops in the network, inefficient routing and even a total collapse of the network. Thus, ensuring secure routing in multihop ad hoc networks is an important consideration.

Internal attacks are especially relevant in ad hoc networks which are operating in hostile environments like enemy battlefields. It is this threat of internal attacks that makes ad hoc security an extremely challenging field. Realize that attacks launched from internal attackers are much harder to detect for two important reasons. First, if a node determines[2] that the routing information that it has received is invalid, it is difficult for it to conclude whether the information that it has received became invalid because of changes in the network topology or because the sending node was compromised. Second, a compromised node would still arguably be able to generate valid signatures using its private keys, thus making it even harder to use cryptography to detect that it has been compromised.

8.2.5 Secure Routing

Ensuring secure routing in multihop ad hoc networks is a big challenge because of the reasons we discussed in the previous section. However, there is one feature of ad hoc data networks which helps in achieving secure routing. This is the existence of multiple (possibly disjoint) paths in these networks which results from each node functioning as a router too. This means that as long as there are a sufficient number of valid (non-compromised) nodes in the network, the routing protocol should be able to bypass the compromised nodes. Approaches taken to achieve secure routing may either be cryptographic or noncryptographic. We will look at some examples of each approach in this section.

An example of a cryptographic approach to secure routing is Authenticated Routing for Ad Hoc Networks (ARAN). ARAN is an on-demand routing protocol which uses a PKC, PKI-based approach where each node in the network has a public key and a private key. A CA is required to issue certificates to all nodes in the system. Each node signs the routing messages using its private key. This allows nodes which

[2] This can be done by comparing the received routing information analytically with the already available routing information.

receive the routing messages to ensure the authenticity of the messages. Signing each routing message prevents against external routing attacks since (arguably) external nodes will not be able to modify signed messages or inject valid new messages. However, this protection comes at the cost of the increased processing overhead that is required by every node for signing every routing message. Also, note that ARAN protects only against attacks from external nodes. ARAN does not protect against attacks from internal nodes which have been compromised.

Another example of a cryptographic approach to secure routing is Security-aware Ad Hoc Routing (SAR)**.** Instead of PKC, it uses Symmetric Key Cryptography (SKC). Each node in the network is assigned a trust level. Also, nodes at each trust level share symmetric encryption keys. A node initiating route discovery can specify the sought trust level for the route; that is, the required minimum trust level for nodes participating in routing. Only nodes at this trust level (which know the correct key) can participate in the routing of this message. Intermediate nodes of different levels cannot decrypt or modify in-transit routing messages. Thus, by specifying the minimum trust level, the initiating node ensures that the message is routed only by nodes which know the secret shared key. Again, this protection comes at the cost of the increased processing overhead that is required by every node for encrypting the routing message. However, the infrastructure overhead may be a little less for this scheme because of the use of SKC as opposed to PKC in ARAN.

An example of a noncryptographic approach to ensure secure routing is suggested by Sergio Marti et al.[3] They propose the use of watchdogs and pathraters in the ad hoc network. The watchdog identifies misbehaving nodes while the pathrater avoids routing packets through these nodes. When a node forwards a packet to the next hop, it also takes on the role of a watchdog to verify that the next node in the routing path (to whom it forwarded the packet) also forwards the packet correctly. The watchdog does so by listening promiscuously to the next node's transmission. If the next node does not forward the packet correctly,[4] then we can conclude that it is misbehaving and has therefore probably been compromised. Thus, each node in the network takes on the role of a network to ensure that the next hop is routing correctly. Note that the use of this approach assumes that per-link encryption is not being used since use of per-link encryption would not allow a watchdog to listen promiscuously to the packets transmitted from other nodes. The information collected by the watchdogs is used by pathraters.

[3] *Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks*, Marti et al.

[4] This scheme can be extended so that the system is tolerant of up to *N* packets not being forwarded by a misbehaving node so as to accommodate the dynamic nature of the ad hoc network.

Just like the watchdog, each node in the network also takes on the role of a pathrater at the appropriate time. The pathrater combines the information collected from the watchdog with the routing table information to select the most robust routing links.

Although this is an interesting approach to prevent routing attacks from internal nodes, it is not free from weaknesses. A node may not be able to detect misbehaving nodes because of multiple reasons. The existence of hidden nodes in the wireless medium allows for the possibility of collisions at the watchdog or the receiver (the next routing hop) which may corrupt the information collected by the watchdog. Also, even though this approach may help prevent internal routing attacks which aim to modify routing paths, it does not prevent against internal routing attacks which aim to partition the network. A malicious node can achieve this by reporting false information from its watchdog. More sophisticated attacks can be launched by colluding compromised nodes.

8.3 Key Establishment and Authentication

As we have seen, key establishment and authentication are the building blocks of network security. Also, these two are also probably the toughest problem in network security. In the following two sections we look at the basic concepts of threshold cryptography which forms the basis of most key establishment and authentication schemes that are being discussed for multihop ad hoc networks.

8.3.1 Threshold Secret Sharing

One of the most prominent solutions to the problem of key establishment and authentication is the use of certificates. Any two nodes in a network may secure (provide confidentiality, data integrity, authentication and nonrepudiation) their communication using certificates. However, issuing and validating certificates requires the deployment of public key infrastructure (PKI) as discussed in Section 2.2.3. The use of PKI relies on a trusted third party (the certificate authority (CA)) to verify the identity and authenticity of other nodes. Therefore, the use of PKI and PKC helps create a trust model in the network where all nodes inherently trust the CA. Note that this means that if a node trusts the CA, it will also trust the identity of another node if the CA verifies this identity.

Let's do a short recap of the role of CA in a PKI. Supposing that Bob wants to talk to Alice using PKC, the following sequence takes place:

1. Bob asks the CA for Alice's public key.

2. The CA responds back with a certificate of the form K_{iCA}\{Alice's public key is K_{wA}\}. In other words, the CA sends the message "Alice's public key is K_{wa}" encrypted with its own private key.
3. When Bob receives this message, it uses the CA's public key (K_{wCA}) to decrypt the certificate and obtain Alice's public key.

The trust model in the system is this: since CA's private key is known only to the CA, no one can forge the certificate and claim another key as Alice's public key. This allows Bob to obtain Alice's public key securely. Once Bob has Alice's public key, he can easily authenticate any node claiming to be Alice by issuing a challenge (RAND) and checking the received response (SRES = K_{iA}(RAND)) using Alice's public key (Is RAND = K_{wA}(K_{iA}(RAND)?). Note that the property of the CA which makes it the trusted node is that only the CA knows its own private key K_{iCA}. Therefore, the security of the whole system is based on ensuring that the K_{iCA} is known only to the CA. We can therefore also refer to this private key, K_{iCA} as the system secret.

Since the PKI by definition requires the existence of infrastructure which is unavailable in ad hoc networks, the threshold secret sharing approach tries to adapt the PKI model to an ad hoc environment by creating a virtual certificate authority. In ad hoc networks since there is no single CA which is always accessible,[5] what is needed is a virtual CA. This virtual CA is formed by distributing the CA's functionality to each local neighborhood. This noncentralized approach also has the advantage that there is no single point of security compromise.[6]

Unfortunately, the role of distributing the CA among multiple physical entities is easier said than done. Realize that the CA is characterized by the possession of the system secret, K_{iCA}. In our distributed-CA model, who would possess this system secret? A trivial solution is to have each of the S nodes which form the virtual-CA possess the system secret. However, this approach has several problems. By having each of the S nodes posses the system secret, we have effectively created multiple instances of the same CA and not a distributed CA as we had intended. This approach also compromises the system secret since it is available to multiple nodes and therefore more vulnerable to compromise.

To achieve a virtual CA, we turn to threshold cryptography, also known as threshold secret sharing, which works by distributing trust among multiple nodes. In this

[5] Or at least not always easily and timely accessible.
[6] Note that by distributing the role of a CA, the scalability problems of a centralized approach are also resolved.

approach, the system secret is divided[7] into Q parts such that any S (< Q) of these parts are enough to carry out a cryptographic operation that would have been possible with the system secret. Note that to carry out a cryptographic operation at least S parts of the system secret are required. A system employing threshold cryptography is therefore defined by the use of two parameters: Q and S. Q nodes posses shares of the system secret and any S of these nodes can work in coalition as a CA. This means that the system can tolerate a compromise of up to S-1 nodes without the security of the whole system being compromised.

We now describe how threshold cryptography is extended to build a virtual CA in an ad hoc environment. We first divided the system secret, K_{iCA} (the private key of the CA) into Q secret shares (k_1, k_2, …., k_Q). A single share of the system secret by itself cannot be used to provide any CA service. However, if S (< Q) such shares are combined, they can be used to provide CA services. Each of these shares is assigned or distributed[8] to a server.

The term *server* is used to refer to a node which will participate in forming the virtual CA. Servers in an ad hoc data network have the following special properties:

1. A server can be initialized securely with its share of the system secret which allows them to act as the server.
2. A server knows the public keys of all nodes which can join the ad hoc network.

Now, consider an ad hoc network where node A wishes to communicate with node B securely. To do so, A needs to authenticate B. A could simply use a challenge-response system with PKC as follows:

1. A sends a challenge (random number) to B
2. B encrypts the challenge with its private key (K_{iB}) to generate a response and sends it back to A.
3. A decrypts the response with B's public key (K_{wB}) and compares the decrypted value with the challenge and if the two match, A concludes that it is communicating with B.

The security of this system lies in the fact that A reliably knows the public key of B. In a PKI, this is achieved by using a signed certificate from the CA. In ad hoc

[7] There are various approaches to achieve this division but we do not go into the details for reasons of brevity.
[8] There is an interesting initialization problem which we will discuss later on.

networks using threshold cryptography, when A needs to find out the public key of B, it sends out a broadcast message to its neighbors requesting a certificate for B. Each server which hears this message generates a partial certificate with its partial system secret k_x and sends it to a combiner. A combiner is a server which takes on the responsibility of combining S partial certificates and generates a complete certificate. Any server can take on the role of a combiner. A server does not require any extra capabilities to be a combiner. Conversely, a server does not gain any extra information about the system secret by being a combiner. Once the combiner has generated the complete certificate by combining S partial certificates, it can send the certificate to A.

Now, let's look at the security of an ad hoc network using threshold cryptography to implement a virtual CA. What happens if a server in the network is compromised? This server can then be used by an adversary to generate an incorrect partial signature. When the combiner uses this invalid partial certificate to generate a complete certificate, it would obviously lead to the complete certificate being invalid. Fortunately, the public key of the virtual CA (K_{wCA}) is known to all nodes in the system.[9] The combiner can therefore use the public key to verify the validity of the complete certificate that it has generated. This can be done, for example, by decrypting the certificate (which has been encrypted using K_{iCA}) using K_{wCA} and verifying that the information in the certificate is correct. If the combiner determines that the complete certificate is invalid, it can use another set of S partial certificates to generate a valid complete certificate. This means that as long as the combiner has access to at least S valid partial signatures it would be able to generate a valid complete certificate. For this reason, the value of S should not be too large. Note that if S (or more than S) servers are compromised, the security of the whole system is compromised. For this reason the value of S should not be too small. These two constraints make the value of S an engineering trade-off.

Consider what happens, however, if the combiner itself is compromised. This is a much more potent threat since it is the combiner which is finally responsible for combining the partial certificates and issuing the complete certificate. A compromised combiner can therefore inject invalid certificates into the system. One solution is to assign the role of a combiner to a server which is more secure than other nodes in the system and thus has a lower probability of being compromised. Since this is not always possible in an ad hoc environment, another approach is to use multiple

[9] That the public key of the CA is well known to all nodes in the system is an underlying assumption of every PKI system.

combiners. In this scenario each combiner issues a complete certificate using its set of partial certificates. The nodes in the system have now multiple sources to get the certificate they want and can use a majority-based scheme to ensure the validity of a certificate.

To protect against attacks where an adversary may compromise multiple servers over a long period of time, the use of secret share updates has been proposed. In this approach, the secret share of each server has to be periodically updated in collaboration with other servers in the system. Since the secret share's validity is limited in time, the adversary must compromise enough servers within a period of finite time to launch a successful attack.

The use of threshold cryptography to create a virtual CA makes two important assumptions regarding system initialization. First, it is assumed that Q servers can be initialized securely with their respective shares of the system secret. Second, it is assumed that each server can be configured securely with the public keys of all nodes which can potentially join the ad hoc network. Both these assumptions basically boil down to the single assumption that the servers can be initially configured over a secure channel. This important assumption can sometimes act as a limitation in providing security in ad hoc networks.

One approach which has been proposed to reduce the dependency of the system on this assumption is localized self initialization. In this approach we still require that the first Q servers be initialized over a secure medium. However, once the first Q servers have been initialized, they can then collaborate to elect new servers. This is achieved by having at least S servers use their secret share (k_x) to generate a partial secret share (ss_x). These partial secret shares are then combined to generate a new secret share which can be assigned to the node which is being initialized as a server.

Let's do a short recap of how a virtual CA works in ad hoc networks. As is true in any PKC system, each node in the ad hoc network has a private-key, public-key pair which it uses to secure communication with other nodes. To certify its keys, each node X, must have a valid certificate issued by the CA of the form $K_{iCA}(X, K_{wX}, T_{sign}, T_{expire})$. This certificate basically says that the CA certifies (by signing the certificate using K_{iCA}) that the public key of node X is K_{wX} and this key is valid between times T_{sign} and T_{expire}. Such certificates which are signed using the system secret (K_{iCA}) are inherently trusted by all nodes in the network. It is these certificates which are then used to provide various security features in the network. So, the aim of a virtual CA is to issue certificates signed using the system secret. The virtual CA is implemented as multiple physically separate nodes (servers) none of which knows the system secret (K_{iCA}) but each one of them knows a share of the system secret. When a node wants a certificate, it sends out a broadcast request. The servers then co-operate to supply the certificate thus providing security in the system.

8.4 Confidentiality and Integrity

In the last section, we discussed how key establishment and authentication may be provided in multihop ad hoc networks. These two security services form the backbone of providing security in any network. Once two nodes in a network have authenticated each other and securely established a security context (that is, securely established keys), encryption and integrity algorithms can be used to secure communication. This part of system security is relatively simple. What is needed is the selection of algorithms and modes suitable for the environment in which the network is expected to operate.

Since the nodes in an ad hoc network environment usually have limited processing power and limited battery lifetimes, most ad hoc networks would prefer a stream cipher for encryption and an integrity algorithm which is not too computation intensive. There are many stream ciphers to choose from as long as we keep in mind that there are some caveats while using stream ciphers in a wireless environment (as WEP demonstrated). See Chapter 5 for more on this.

8.5 Bluetooth

One of the most popular ad hoc standards today is Bluetooth. Some of the salient features of Bluetooth are as follows:

- Wireless ad hoc networking technology.
- Operates in the unlicensed 2.4 GHz frequency range.
- Geographical coverage limited to personal area networks (PAN).
- Point-to-point and point-to-multipoint links.
- Supports synchronous and asynchronous traffic.
- Concentrates on single-hop networks.
- Frequency hopping spread spectrum (FHSS) with gaussian frequency shift keying (GFSK) modulation at the physical layer.
- Low power and low cost given important consideration.
- Adopted as the IEEE 802.15.1 standard for physical layer (PHY) and media access control (MAC) layers.

The Bluetooth standard limits its scope by dealing only with single-hop ad hoc networks with limited geographical coverage (PAN). In the previous sections we saw that multihop ad hoc networks present a unique set of challenges which are still an active area of research. The Bluetooth standard brings ad hoc networks to the commercial forefront by concentrating on single-hop PAN ad hoc networks. Removing the multihop feature from ad hoc networks makes things a lot simpler.

The Bluetooth Special Interest Group (SIG) was founded in 1998 with the aim of developing Bluetooth as a short-range wireless inter-connectivity standard.[10] In other words, Bluetooth deals with ad hoc networks whose geographical coverage is limited to PAN. Typical applications of Bluetooth today include connecting a wireless headset with its cell phone, interconnecting the various components (keyboard, mouse, monitor, and so on) of a PC, and so on.

Before we get into the details of Bluetooth and its security, it is important to emphasize that Bluetooth is by no means the only ad hoc network standard. Another popular ad hoc standard is 802.11 in its IBSS mode.

Since Bluetooth networks have been so commercially successful, we briefly look at Bluetooth security in this section.

8.5.1 Bluetooth Basics

A typical Bluetooth network, called the piconet, is shown in Figure 8.2. Each piconet has one master and can have up to seven slaves.[11] Therefore, there can be at most eight devices in a piconet. A slave can communicate only with the master and a master can obviously communicate with any of the slaves. If two slaves wish to communicate with each other, the master should relay this traffic. In effect, therefore, we have a logical star topology in a piconet, with the master device at the center.

Comparing the piconet to a 802.11 network, the piconet is the equivalent of a BSS (though with a much smaller geographical coverage), the master device is the equivalent of the AP (except that it is not connected to any distribution system) and the slave devices are the equivalent of the Stations (STAs).

[10] The Bluetooth standard is also being accepted as the IEEE 802.15 standard.

[11] To be precise, a piconet has one master and up to seven active slaves. There is no limit on the number of slaves in a piconet which are in "park" or "hold" state. This distinction is irrelevant from a security perspective however.

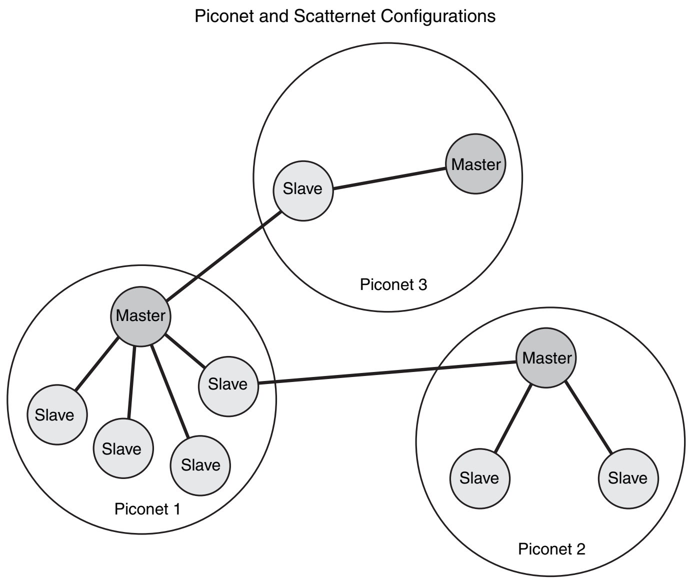


Figure 8.2: Bluetooth Networks

A Bluetooth device may participate in more than one piconet simultaneously, as shown in Figure 8.3. In such a scenario, it is possible for the devices in two piconets to communicate with each other by having the common node act as the bridge and relay the inter-piconet traffic. The two piconets are now joined together and form a scatternet. Even though scatternets are theoretically possible, they are rare in commercial deployments since they pose tough practical problems like routing and timing issues. The Bluetooth standard concentrates mostly on single-hop piconets and we limit our discussion to piconet security. Scatternets (and their security) are an active area of research and involve a lot of the security issues that we discussed before Section 8.5.

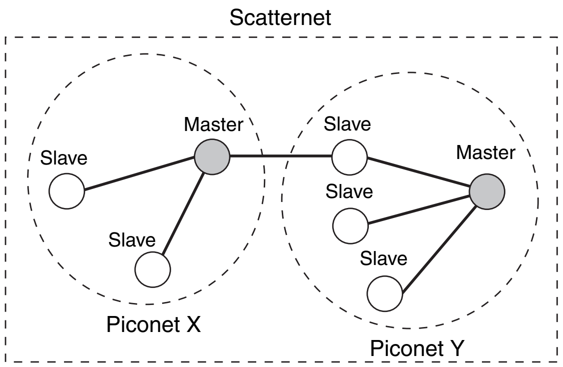


Figure 8.3: Piconets and Scatternets in Bluetooth

8.5.2 Security Modes

Just like IEEE 802.11 standard, the Bluetooth standard also defines Layer 1 and Layer 2 of the OSI stack to achieve communication in single-hop personal-area ad hoc networks. However, by their very nature, ad hoc networks (Bluetooth) are a much less controlled environment than WLANs (802.11). This, combined with the fact that the Bluetooth standard may be used by a wide range of applications in many different ways, makes interoperability a much bigger challenge in Bluetooth networks. To ease the problem of interoperability, the Bluetooth SIG defined application profiles. A profile defines an unambiguous description of the communication interface between two Bluetooth devices or one particular service or application.

There are basic profiles which define the fundamental procedures for Bluetooth connection and there are special profiles defined for distinct services and applications. New profiles can be built using existing profiles, thus allowing for a hierarchical profile structure as shown in Figure 8.4.

Each service or application selects the appropriate profile depending on its needs, and since each application may have different security requirements, each profile may define different security modes. The most fundamental profile is the Generic Access Profile (GAP) which defines the generic procedure related to the discovery of the Bluetooth devices and link management aspects of connection between them. The GAP defines three basic security modes of a Bluetooth device.

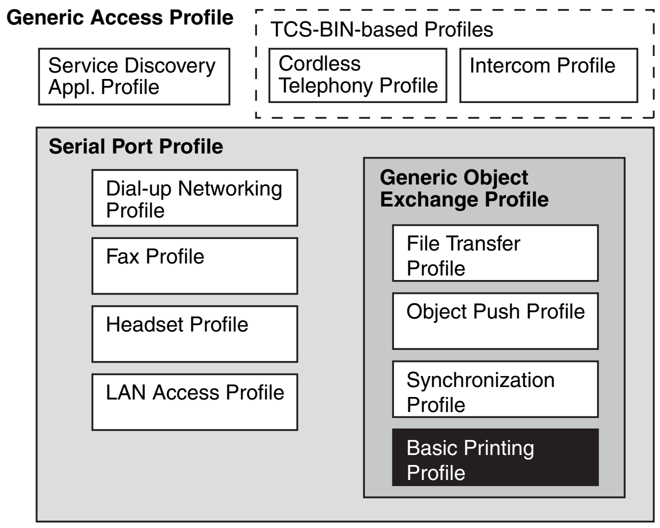


Figure 8.4: Profiles in Bluetooth

Before we discuss the different security modes, it is important to keep a few things in mind. First, the security mechanisms (authentication and encryption) specified by the Bluetooth standard are implemented at the link layer (Layer 2). This means that the scope of Bluetooth security is the Layer 2 level link between two nodes separated by a single hop. To be explicit, Bluetooth security does not deal with end-to-end security[12] and does not deal with application layer security. If such security mechanisms are required, they have to be arranged for outside the scope of the Bluetooth standard.

Second, all Bluetooth devices must implement an authentication procedure: that is a requirement.[13] Bluetooth devices may or may not implement encryption procedures: that is optional. However, just because a device implements or supports authentication and/or encryption, does not mean that this device would use these security features in a connection. What security features are used for a Bluetooth connection depends on the security modes of the master and the slave in the connection. Table 8.1 summarizes what security features are used in a Bluetooth connection depending on the security mode of the master and the slave.

Table 8.1

Security Modes **Master Slave**	**1**	**2**	**3**

1	**Authentication** = No;
	Encryption = No;
		Authentication = if the master *app.* demands it;
		Encryption = if the master *app.* demands it;
			Authentication = Yes;
			Encryption = if the master *policy* demands it.

2	**Authentication** = if the slave *app.* demands it. Encryption = if the slave *app.* demands it.	**Authentication** = if the master or the slave *app.* demands it; Encryption = if the master or the slave *app.* demands it;	**Authentication** = Yes; Encryption = if the master *policy* or if the slave *app.* demands it.
3	**Authentication** = Yes. Encryption = if the slave *policy* demands it;	***Authentication*** = Yes. Encryption = if the slave *policy* or if the master *app.* demands it.	**Authentication** = Yes. Encryption = if the master or the slave *policy* demands it;

[12] The source and destination nodes may be more than one hop away as in a scatternet.
[13] On the other hand, implementing encryption procedures is optional.

Security mode 1 is the unsecured mode in Bluetooth. A device using this mode does not demand authentication or encryption from its peer at connection establishment. This means that a device in security mode 1 offers its services to all devices which wish to connect to it. However, if the communicating peer device (say B) wishes to authenticate a node which is using security mode 1 (say A), then A must respond to the challenge that B sends since A as a Bluetooth device must support authentication. Similarly, if B wishes to use encryption on its link with A, then A must turn on its encryption if it supports it.

On the other end of the spectrum is security mode 3 which is the always-on security mode in Bluetooth. A device which is in security mode 3 shall always initiate an authentication procedure. This means that this device will not communicate with a device unless it can authenticate it. The encryption part in security mode 3 is not as simple. Recall that the Bluetooth standard does not require every device to implement encryption. If a device in security mode 3 (say A) is trying to communicate with a peer which implements encryption, then this is not an issue since the link can be secured. However, if A wishes to communicate with a peer which does not implement encryption (say, B), then we have a problem. How this situation is to be handled is left to higher layers (in other words, the security policy manager). The security manager may decide not to communicate with B, or it may decide to communicate with B without using encryption.

Security mode 2 lies in between modes 1 and 3 and has been designed to offer applications flexibility. Whether or not authentication and/or encryption is used, is left to the decision of the security policy manager. This is achieved by relinquishing control of security use to a higher layer security manager. In other words, security mode 2 works by using service level-enabled security. This mode is most useful in scenarios where multiple applications (with different security requirements) may run on a single Bluetooth device. The security manager can then co-ordinate what security policy to use depending on the application which is running.

In summary, whether or not authentication and/or encryption is used in a Bluetooth communication link depends on a lot of factors: the security needs of the application, the security capabilities of the devices, the security mode of the device and the security (access) policies. The security manager considers all these factors when deciding if (and at which stage of connection establishment) the device should start using the security procedures.

8.5.3 Key Establishment

Key establishment is probably the most complex part of Bluetooth security. A large part of this complexity is in the key hierarchy because of the large number of keys involved in the Bluetooth security process. Figure 8.5 shows a classification of most of the keys involved in the Bluetooth security process. The key hierarchy in Bluetooth varies a little bit depending on whether we are dealing with unicast communication (between two devices) or broadcast communication. For the rest of this section we assume that we are dealing with unicast communication. Finally, in Section 8.5.3.7 we will point out how broadcast communication key hierarchy is different from unicast key hierarchy.

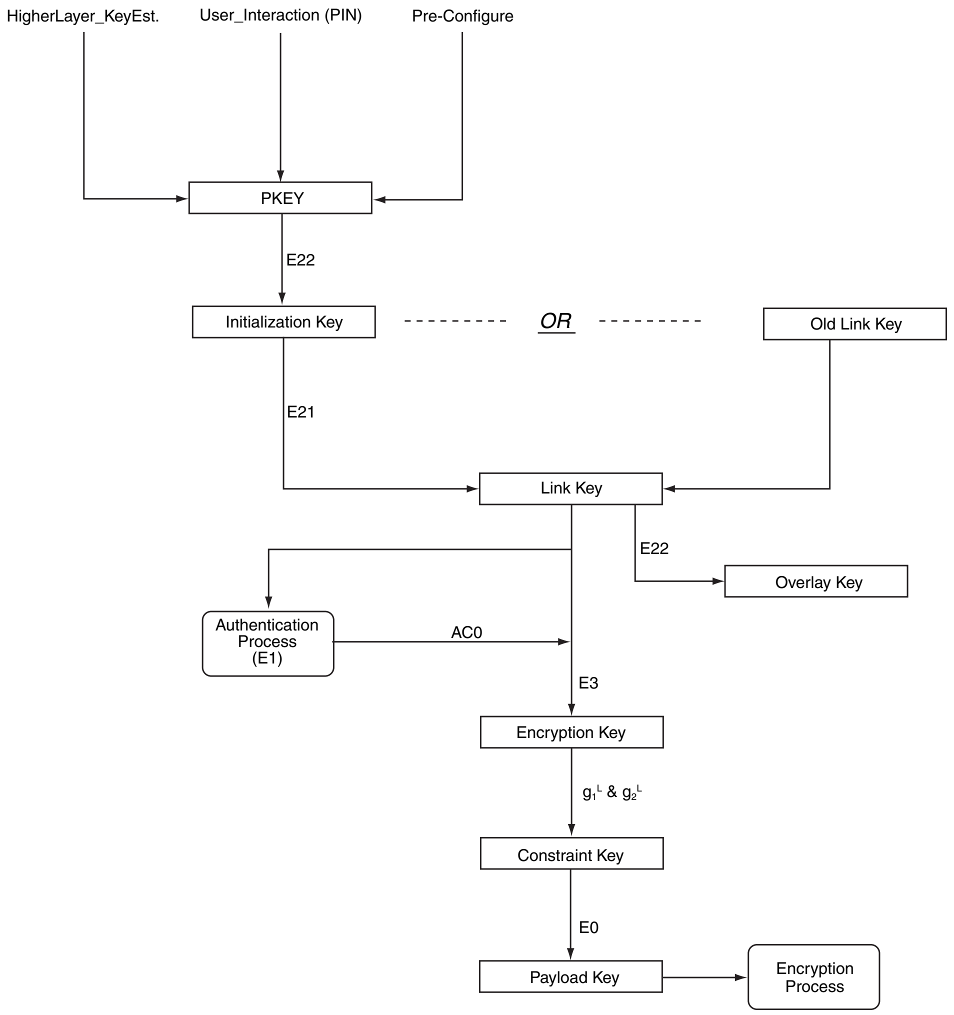


Figure 8.5: Bluetooth Key Hierarchy

8.5.3.1 Pass Key

At the top level is the Pass-Key (PKEY). The PKEY is basically the shared secret between the two communicating devices. There are two types of PKEYs: variable PKEYs and fixed PKEYs. Variable PKEYs refer to PKEYs that can be chosen at the time of the "pairing" (the process by which two Bluetooth devices establish a shared secret that they can use for securing communication). This is usually achieved by prompting the user to enter a PKEY during the pairing procedure. Obviously, users of both devices should enter the same PKEY. On the other hand, fixed PKEYs refer to PKEYs that are preconfigured into the Bluetooth device. Again, both the communicating devices should be preconfigured with the same PKEY. Even though variable PKEYs are more secure (since the PKEY can be changed on a per-pairing basis), both variable and fixed PKEYs serve specific purposes. Consider for example, a scenario where users in a conference room wish to form a Bluetooth network using their laptops. Such a scenario is well-suited for using variable PKEYs, since each device has user interaction capabilities. On the other hand, consider the Bluetooth network between the headset and its cell phone. The Bluetooth headset must use a fixed PKEY since there is no[14] user interaction capability on the headset. The Bluetooth standard also allows the use of higher layer key-establishment protocols to generate the PKEY and pass it on to the Bluetooth stack.

Since the PKEY can come from one of many sources, instead of specifying the exact length of the PKEY, the Bluetooth standard specifies that the PKEY can be as long as 128 bits. This allows for devices prompting the user for a PKEY to enter a much smaller PKEY (or a PIN) thus making user interaction a little more convenient. However, using a smaller PKEY has other drawbacks. As we will see in the next few sections, the PKEY is the starting point for establishing the Link Key, which in turn forms the basis of all security in Bluetooth. To be precise, the PKEY is the shared secret between the two communicating endpoints that ensures the Link Key is known only to the communicating end-points. The use of a smaller PKEY means that an attack like the dictionary attack becomes much easier to launch. In this context, a dictionary attack involves calculating all the link keys that can be derived from all possible PKEYs. This list of link keys is maintained in a table and can then be used against <plaintext, ciphertext pairs>[15] to determine which link key is being used.

[14] Or rather hardly any.

[15] For example <AU_RAND, SRES> pairs.

8.5.3.2 Initialization Key

The next level in the hierarchy is the Initialization Key (IK or K_{INIT}). The K_{INIT} is a short-lived temporary key that is used (and exists only) during the pairing process when two Bluetooth devices start communicating for the first time. The K_{INIT} is derived using the E_{22} algorithm and three inputs: PKEY, IN_RAND and L_{PKEY}. The PKEY is the pass-key that we just talked about and L_{PKEY} is the length of this pass-key in bytes. Finally, the IN_RAND is a 128-bit random number generated by the device. The process of deriving the IK from these inputs is as follows:

PKEY' = PKEY + padding bits.
$L_{PKEY}' = \min(L_{PKEY} + 6, 16)$.
$K_{INIT} = E_{22}(PKEY', IN\_RAND, L_{PKEY}')$

The need for padding bits stems from the flexibility of allowing the PKEY to be as long as 128 bits but not specifying the exact length of the PKEY. The padding shown above is needed to ensure that PKEY is exactly 128 bits and these padding bits come from the claimant's[16] BD_ADDR (the 48-bit MAC address).

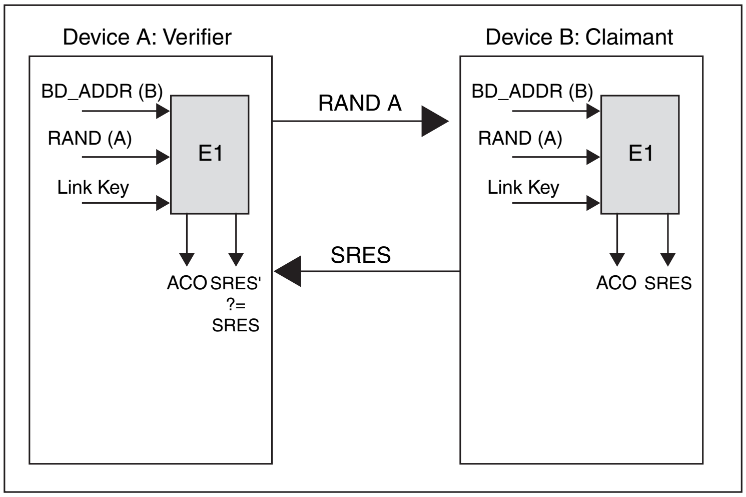


Figure 8.6: Bluetooth Authentication

8.5.3.3 Link Key

The next level in the key hierarchy is the Link Key (LK). The link key is the shared secret established between the two Bluetooth devices when the pairing sequence ends.

[16] Bluetooth uses the terms claimant and verifier to refer to the two Bluetooth devices which are involved in the communication. Claimant refers to the device which is claiming an identity and verifier refers to the device which verifies this claim.

There are two types of link keys specified by the Bluetooth standard: the unit key and the combination key. The use of unit key has now been deprecated because of the security loopholes that result from its use. We therefore concentrate only on the combination link keys here. The terms link key and combination key from now on are used interchangeably throughout the text to refer to the same key. The combination/link key is derived either from the existing link key (if such a key exists) or the initialization key, K_{INIT} (if no link key exists between the two devices). We will see how the combination link key is generated but note that we talk about generating a link key from the existing link key! What does this mean?

Consider two devices which establish a combination/link key and communicate securely for a while. When the communication is terminated, what should the two devices do with the link key which they used for this session? One approach is to discard these keys. This approach requires that every time these two devices want to establish a Bluetooth session, they must generate the link key from scratch. Even though cryptographically this is a more pure approach, in the interest of efficiency Bluetooth allows devices to store the link keys that they generate in nonvolatile memory and reuse this link key for future communications with the same device. In other words, unlike the initialization key, the link key is a semi-permanent key. Therefore, each device may[17] maintain a database of <remote_device_address, link_key> pairs for the link keys it wishes to reuse. Such an approach is specially suited to devices which repeatedly connect to a small fixed set of devices for example the Bluetooth headset which usually connects to its cell phone.

Note that just because devices can reuse link keys does not mean that they should never change the link key. Periodically changing the link key is a recommended practice since, as we know by now, the more a key is used or exposed, the more it is vulnerable to compromise. It is in these scenarios that a link key is used to generate another link key. In other scenarios where two devices do not already have a link key shared between them, the K_{INIT} is used to generate the link key. In summary, the end of the pairing process in Bluetooth should lead to the establishment of a link key which the two devices can use for securing their communication. This link key (combination key) can come from three sources:

1. Use an existing link key that the two devices had established previously.
2. Use an existing link key to generate a fresh link key.
3. Use the initialization key, K_{INIT} to generate a link key.

[17] The decision whether or not to store a particular link key in the database may be left to the user.

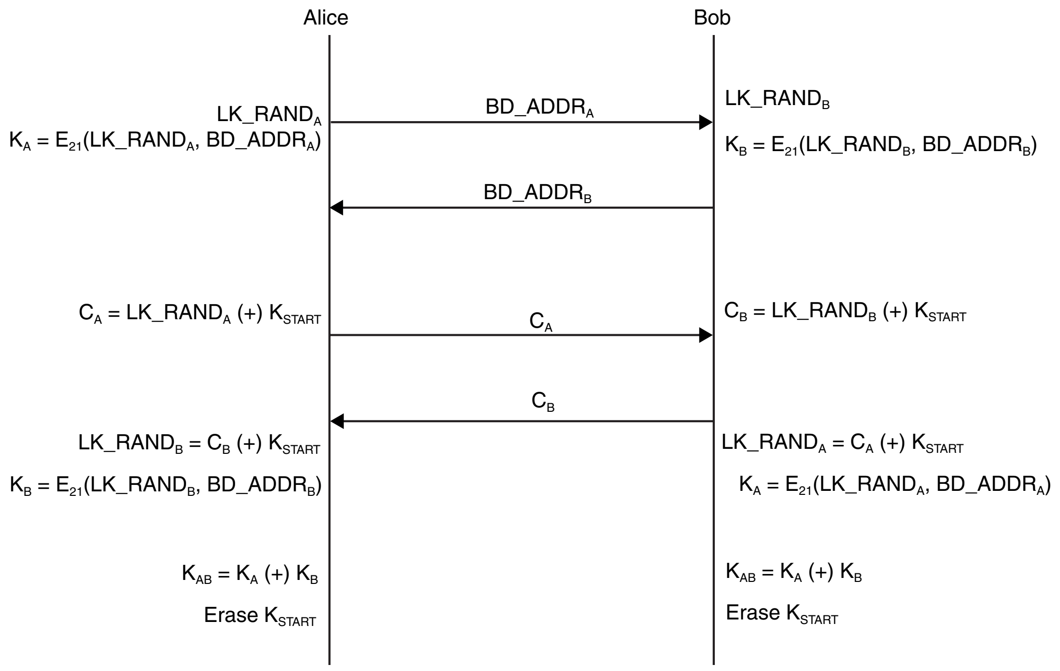

Alice
Bob
LK_RAND$_A$
$K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A)$
BD_ADDR$_A$
LK_RAND$_B$
$K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B)$
BD_ADDR$_B$
$C_A = LK\_RAND_A$ (+) K_{START}
C_A
$C_B = LK\_RAND_B$ (+) K_{START}
C_B
$LK\_RAND_B = C_B$ (+) K_{START}
$K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B)$
$LK\_RAND_A = C_A$ (+) K_{START}
$K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A)$
$K_{AB} = K_A$ (+) K_B
Erase K_{START}
$K_{AB} = K_A$ (+) K_B
Erase K_{START}

Figure 8.7: Bluetooth Key Establishment

The process of generating the link key is as follows. We start with either the existing link key or the K_{INIT} depending on the context. Let us call this starting key K_{START}.[18] The most important property to remember about K_{START} is that it is shared secretly between the two communicating devices; that is, it is known only to the two communicating devices. Now, each of the communicating devices (say A and B) generate a private key using the E_{21} algorithm, their BD_ADDR and a self-generated random number (LK_RAND). Therefore,

$$K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A)$$

$$K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B)$$

The combination Link Key is simply the XOR of K_A and K_B. However, the process of establishing the Link Key is not yet complete since K_A is available only at A and K_B is available only at B. What is needed is a way for B to be able to generate K_A and for A to be able to generate K_B.

[18] Therefore K_{START} = LK or K_{INIT}.

To be able to generate K_B, A needs to know LK_RAND_B and BD_ADDR_B. Arguably, A already knows BD_ADDR_B since it is going to communicate with it. So, what is needed is a secure way to get LK_ADDR_B from B to A. Here is where K_{START} comes in. B XORs LK_ADDR_B with K_{START} and sends it to A. Since K_{START} is a shared secret known only to A and B, we are assured that this transmission of LK_ADDR_B is secure. Once A gets to know $LK\_ADDR_{B,}$ A can generate K_B too. Following the exact same procedure, B can generate K_A too. At this point both A and B have calculated K_A and K_B and can therefore easily calculate the combination link key as K_A XOR K_B.

Note from Figure 8.7 that after the establishment of the new combination link key, K_{START} is deleted by both endpoints since the new combination link key should be used from now on.

8.5.3.4 Encryption Key

The combination link key is used in the authentication process (see Section 8.5.4) as we will see in the next section. The link key is also used for generating the ciphering key (CK or K_C) which is the next key in the key hierarchy. The K_C is derived from the link key using the E3 algorithm as follows:-

$$K_C = E_3(K, EN\_RAND, COF)$$

The link key is denoted by K. EN_RAND refers to a 128-bit random number and COF refers to a 96-bit Ciphering Offset. The value of COF is equal to the value of the Authentication Ciphering Offset (ACO), which is derived from the authentication process. (See Section 8.5.4 for details on this.)

8.5.3.5 Constraint Key

The next step in the key hierarchy is the constraint key (K_C'), a.k.a. the constraint encryption key. The reason for the existence of this key is export restrictions. Many countries impose restrictions on the export of encryption hardware. Specifically, hardware which is capable of encrypting above certain key strengths is not exportable. For this purpose, Bluetooth put in a key strength constraining mechanism that reduces the 128-bit K_C to a 128-bit K_C' whose effective key length (key strength) can be any value less than 128 bits. The derivation of K_C' from K_C is achieved in hardware using linear feedback and feed forward registers and can be given as:

$$K_C'(x) = g_2^L(x)\{K_C[\mathrm{mod}\ g_1^L(x)]\}$$

Where L is the desired effective length and g_1 and g_2 are two polynomials specified by the Bluetooth standard.

8.5.3.6 Payload Key

Finally, the Payload Key (P_K) is the actual key that is used to encrypt (decrypt) Bluetooth packets. Therefore, P_K is derived from the Constraint Key K_C' using the E_0 algorithm as follows:

$$K_P = E_0(K_C', \text{CK\_VAL}, \text{BD\_ADDR}, \text{EN\_RAND})$$

Where BD_ADDR is the 48-bit Bluetooth (read MAC) address of the device, EN_RAND is a 128-bit random number and CK_VAL is the 26 bits of the current clock value (bits 1–26 of the master's native clock).

8.5.3.7 Broadcast Key Hierarchy

All our discussion regarding Bluetooth key hierarchy has assumed that the Bluetooth communication is between two devices (a master and a slave). However, the Bluetooth standard also supports broadcast communication where a master may broadcast data to all its slaves. Realize that a broadcast transmission is different from multiple unicast transmissions. In a broadcast transmission, the master sends out a message to a special broadcast address and all slaves in the piconet accept this message. In the latter case, a master sends out multiple copies of a message to each slave individually. In this section, we talk about broadcast in the former sense.

Recall from Section 8.5.3.3 that the (combination) link key in unicast communication in a piconet was generated using the addresses and random numbers from both endpoints involved in the communication. In broadcast communication, this becomes a dilemma since communication is not between two endpoints. Therefore, the link key is replaced by the use of a Master Key (K_{master}). The master key is derived independently by the master without involving any of the slaves. This is done using the E_{22} algorithm as follows:

$$K_{master} = E_{22}(\text{LK\_RAND1}, \text{LK\_RAND2}, 16)$$

Since the master key is derived only at the master, we need a way to communicate the K_{master} to all the slaves in the piconet securely. This is done using the Overlay Key, K_{ovl}. The overlay key is derived from the current link key as follows:

$$K_{overlay} = E_{22}(K, \text{RAND3}, 16)$$

Since the master and each of the slaves in a Bluetooth piconet share a link key, the overlay key can be securely established between the master and each of the slaves. This overlay key can then be used for conveying the master key to each of the slaves. Finally, the master key is used for securing broadcast communication in a piconet.

Note that unlike the link key which is a semi-permanent key (stored in nonvolatile memory for future use), the master key is a temporary key which is never stored in nonvolatile memory (and never re-used).

8.5.3.8 The Algorithms

The key hierarchy that we have discussed in the previous sections uses five algorithms: E_0, E_1, E_3, E_{21} and E_{22}. Of these five algorithms, four (E_1, E_3, E_{21} and E_{22}) are based on a block cipher and one on a stream cipher (E_0). The discussion of E0 as a stream cipher is postponed to Section 8.5.5 where we see how Bluetooth packets are encrypted. To understand the use of the block cipher for four of these algorithms, recall that all keys in Bluetooth have a length of 128 bits. Using a 128-bit block cipher in key derivation means that we can feed one key directly as input into the block cipher to generate the key of the next level in the hierarchy. All these four algorithms use the same underlying block cipher: SAFER+.

At the time the Bluetooth standard was being ratified, the National Institute of Standards and Technology (NIST) was considering contenders for the Advanced Encryption Standard (AES). SAFER+ was one of the strong contenders for AES which had been very thoroughly cryptanalyzed. Bluetooth therefore chose SAFER+ as the block cipher to be used for the implementation of E_1, E_3, E_{21} and E_{22}. The details of this algorithm are not discussed here for reasons of brevity and the interested reader is referred to the Bluetooth standards.

8.5.4 Authentication

The authentication process always involves two endpoints: the claimant (which claims a certain identity) and the verifier (which wishes to verify that the claimant is actually the identity it is claming to be). In the Bluetooth piconet context, the roles of claimant and verifier are orthogonal to the rule of the master and the slave. In other words, either the master or the slave can act as the verifier. Who is the verifier depends on higher layers. The application or the user who wishes to ensure the identity of the remote end (and who therefore starts the authentication process) takes on the role of the verifier. For mutual authentication, both end-points take on the role of the verifier one at a time. Figure 8.8 shows a mutual authentication process in Bluetooth.

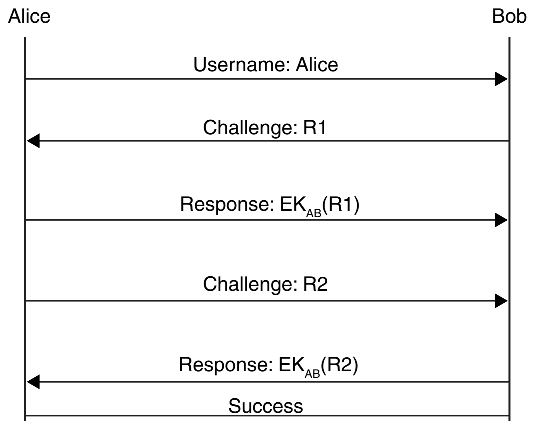


Figure 8.8: Bluetooth Mutual Authentication

The authentication process in Bluetooth is basically a challenge-response protocol which is carried out using the existing link key. Consider a device (claimant) which initiates communication with A claiming to be B. A wishes to verify that the claimant is in fact B. To verify this A sends the claimant a random number, AU_RAND. On receiving this, the claimant is expected to send back a signed response SRES calculated using the E_1 algorithm as follows:

$$SRES = E_1(K, AU\_RAND, BD\_ADDR_B).$$

Since the AU_RAND and BD_ADDR_B may easily be known publicly, the security lies in K, the link key. The underlying assumption of the Bluetooth authentication process is that the link key is known only to the two endpoints which established it (See Section 8.5.3.3 on how this is achieved). Since only B knows the correct K which it established with A, only B would be able to generate the correct SRES. So, all A needs to verify the identity of the claimant is to ensure that the response sent back by the claimant is equal to SRES.

As Figure 8.8 shows, mutual authentication can also be carried out in Bluetooth with B now taking on the role of a verifier. B sends out a challenge to A and carries on the authentication process to verify the identity of A. The E_1 algorithm used for generating the SRES in the authentication process also produces a 96-bit Authentication Ciphering Offset (ACO) as an output. As we saw in Section 8.5.3.4, this ACO is used for generating the ciphering key, K_C. It is this ACO which "links" the authentication process to the rest of the session. In other words, the ACO serves to link the security context established by the authentication process to the rest of the session.

Note that if mutual authentication is desired, first A acts as the verifier and B as the claimant. Next, the roles are swapped with B acting as the verifier and A as the claimant. Therefore, in mutual authentication, there would be two ACOs that would be produced: one from each authentication process. In such a scenario, the standard specifies that the ACO from the latter authentication process be used for generating the encryption key. Therefore, the security context is linked to the last authentication process that is carried out.

8.5.5 Confidentiality

As we discussed in Section 8.5.3.6, the payload key P_K, which is used for encrypting outgoing messages is derived using the E_0 algorithm. The E_0 algorithm is basically a stream cipher which generates a key stream. This key stream is then XORed with the plaintext of the messages to create the ciphertext. The design of the E_0 stream cipher is not based on any existing stream cipher but is a proprietary algorithm specified by the Bluetooth standard.

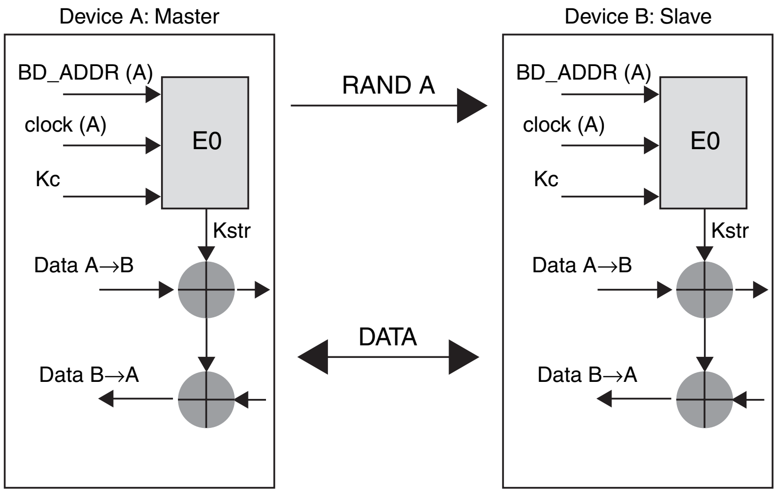


Figure 8.9: Bluetooth Encryption

Figure 8.9 shows the encryption process in Bluetooth networks. From Section 8.5.3.6, we know that P_K is derived from the Constraint Key K_C' using the E_0 algorithm as follows:

$$K_P = E_0(K_C\text{'}, \text{CK\_VAL}, \text{BD\_ADDR}, \text{EN\_RAND})$$

Where BD_ADDR is the 48-bit Bluetooth address of the device, EN_RAND is a 128-bit random number and CK_VAL is the 26 bits of the current clock value (bits 1–26 of the master's native clock). Next, this P_K is fed into the key stream generator. This key stream generator then produces a key stream which is XORed with the plaintext to produce the ciphertext. There are a few important things to note about the encryption process in Bluetooth:

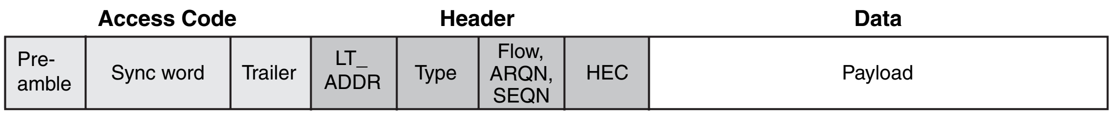


Figure 8.10: Bluetooth Packet Format

First, not all bits of the Bluetooth packet are encrypted. Figure 8.10 shows the format of a Bluetooth packet. It consists of an access code followed by a header and finally the payload. The access code is derived from the BD_ADDR of the master of the piconet and since every piconet has a unique master, the access code uniquely identifies a piconet. The access code is therefore used by the devices in a piconet to determine if a packet is for another piconet, in which case the packet is discarded.

The access code also defines where a slot boundary lies and is therefore also used by the slaves in a piconet to synchronize their clocks to the master's clock. It is therefore not a surprise that the access code in Bluetooth packets is not encrypted. Next, the header in the Bluetooth packet is also not encrypted. The reason for this is also pretty obvious when you consider that the header contains the address of the destination device. This information is obviously needed by all devices in the piconet to determine whether or not a particular packet is intended for it or not. Therefore, the bottom line is that only the payload in the Bluetooth packet is encrypted.

Second, as shown in Figure 8.9, the CRC is appended to the packet before it is encrypted. In other words, the CRC along with the payload is also encrypted.

Third, realize that using a stream cipher in a wireless medium is a security loophole as discussed in Section 7.5.1 (What's Wrong with WEP?). Just like WEP tries to overcome the drawbacks of using a stream cipher by changing the key for each packet, Bluetooth uses the same approach: the P_K is derived for each Bluetooth packet.[19] However, unlike WEP where the per-packet key is calculated simply by prepending the IV with the master key, the derivation of the per-packet key in Bluetooth is much more cryptographically involved, thus making Bluetooth encryption more secure than WEP. Let us take a closer look at Figure 8.9.

As Figure 8.9 shows, the encryption process in Bluetooth can be separated into three distinct blocks. The first block consists of deriving the payload key, P_K. We saw at the beginning of this section how this is achieved. This P_K feeds into the second block and acts as the initialization seed to a key stream generator. In other words, P_K is used to initialize the key stream generator. The key stream generated from the second block feeds into the third block which is nothing but a simple XOR operation between this key stream and the payload[20] of the Bluetooth packet.

Finally, realize that to change the P_K on a per-packet basis, we need a variable which changes on a per-packet basis. One of the inputs required for generating the payload key, P_K is CK_VAL, the lower 26 bits of the master clock. Since the lowest bit of the master clock (and hence the CK_VAL) changes every 625 microseconds, this means the value of the P_K can be derived afresh every 625 microseconds. However initializing the key stream generator with P_K takes some time. This is where guard space comes in. The Bluetooth standard specifies a guard space between the end of the payload in one packet and the start of the next packet. This guard space must be

[19] Multislot packets do not require a change of the payload key when passing a slot boundary within a packet.
[20] Along with the CRC.

at least 259 microseconds, which is enough time for the key stream generator to be initialized. The overall timing sequence is as shown in Figure 8.11 where “running the stream cipher” and “initializing key stream generator” slots alternate.

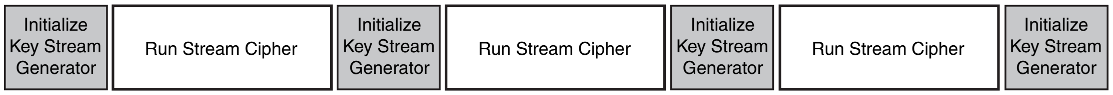

Initialize Key Stream Generator
Run Stream Cipher
Initialize Key Stream Generator
Run Stream Cipher
Initialize Key Stream Generator
Run Stream Cipher
Initialize Key Stream Generator

Figure 8.11: Bluetooth Stream Cipher Periodicity

8.5.6 Integrity Protection

The Bluetooth standard relies on a cyclic redundancy check (CRC) to provide message integrity. Recall from Section 7.6 that WEP also used CRC for providing message integrity. We discussed the loopholes of this approach in Section 7.7 and concluded that using a linear noncryptographic integrity check mechanism like CRC leaves a lot to be desired as far as integrity protection is concerned. We also saw that just encryption does not automatically provide integrity. In other words, just because a message is encrypted does not mean that it can’t be modified in-transit without the receiver’s knowledge. The bottom line is that by choosing CRC, Bluetooth fails to provide any real[21] integrity protection.

8.5.7 Enhancements

As we said at the beginning of this chapter, ad hoc networking technology is still in its nascent stage. There are many issues to be resolved and security is such an issue. The Bluetooth standard is also expected to evolve as ad hoc networking technology evolves, but that may be a topic for the next edition of this book.

[21] That is, any strong enough.

References

[1] *Evaluation and Implementation of Secure Mobile Commerce Systems*, G. Richter.

[2] *IEEE 802.1X for Wireless LANs*, Bernard Aboba et al.

[3] *802.11 Security Series*, J. Walker.

[4] *A Technical Comparison of TTLS and PEAP*, Matthew Gast.

[5] *Unsafe at any Key Size: An Analysis Of the WEP Encapsulation*, J. Walker.

[6] *IEEE Std 802.11, Standards for Local and Metropolitan Area Networks: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[7] *EAP Tunneled TLS Authentication Protocol*, Paul Funk et al.

[8] *Security on Ad Hoc Networks*, Arun Kumar Bayya et al.

[9] *Wireless LANs: The 802.1X Revolution*, Bernard Aboba.

[10] *Using the Fluhrer, Mantin, and Shamir Attack to Break WEP*, Adam Stubblefield et al.

[11] *Providing Robust and Ubiquitous Security Support for Mobile Ad Hoc Networks*, Jiejun Kong et al.

[12] *Ad Hoc Mobile Networks*, Hridesh Rajan.

[13] *Self-securing Ad Hoc Wireless Networks*, Haiyun Luo et al.

[14] *Talking to Strangers: Authentication in Ad Hoc Wireless Networks*, Dirk Balfanz et al.

[15] *Where does Wi-Fi Security Come From?*, Jesse Walker.

[16] *Wireless Privacy: Analysis of 802.11 Security*, Nikita Borisov.

[17] *Securing Ad Hoc Networks*, Lidong Zhou.

[18] *A Cryptographic Evaluation of IPsec*, Niels Ferguson et al.

[19] *GSM (and PSN) Security and Encryption*, Charles Brookson.

[20] *Overview of the Global System for Mobile Communications*, John Scourias.

[21] *Key Agreement in Ad Hoc Networks*, N. Asokan et al.

[22] *On the Security of 3GPP Networks*, Michael Walker.

[23] *Secure Coding: Principles and Practices*, Graff and Wyke.

[24] *Applied Cryptography*, Schneier.

[25] *Firewalls and Internet Security*, Cheswick et al.

[26] *Network Security: Private Communication in a Public World*, Kaufman et al.

[27] *Wireless Personal Communication Systems*, Goodman.

[28] *802.11 Wireless Networks: The Definitive Guide*, Gast.

[29] *Wireless Communications: Principles and Practice*, Rappaport.

[30] *Wireless Communications and Networks*, Stallings.

[31] *The Code Book*, Simon Singh.

[32] *3G Security Principles*, Myagmar and Gupta.

[33] *Radio Interfaces Make the Difference in 3G Cellular Systems*, Malcolm W. Oliphant.

[34] *3G Security Architecture: 3GPP TS 33.102.*

[35] *3G Security Principles and Objectives: 3G TS 33.120.*

[36] *A Guide to 3rd Generation Security: 3G TR 33.900.*

[37] *Security and Denial of Service Threats in GSM Networks*, Valer Bocan et al.

[38] *Routing Protocols for Ad Hoc Mobile Wireless Networks*, Padmini Misra.

[39] *On the Scalability of IEEE 802.11 Ad Hoc Networks*, Huang et al.

[40] *A Performance Comparison of Multihop Wireless Ad Hoc Network Routing Protocols*, Josh Broch et al.

[41] A Comprehensive Review of 802.11 Wireless LAN Security and the Cisco Wireless Security Suite: White Paper.

Index