CHAPTER 8

Cryptography

This chapter presents the following:

- History of cryptography
- Cryptography components and their relationships
- Government involvement in cryptography
- Symmetric and asymmetric key algorithms
- Public key infrastructure (PKI) concepts and mechanisms
- Hashing algorithms and uses
- Types of attacks on cryptosystems

Cryptography is a method of storing and transmitting data in a form that only those it is intended for can read and process. It is considered a science of protecting information by encoding it into an unreadable format. Cryptography is an effective way of protecting sensitive information as it is stored on media or transmitted through untrusted network communication paths.

One of the goals of cryptography, and the mechanisms that make it up, is to hide information from unauthorized individuals. However, with enough time, resources, and motivation, hackers can break most algorithms and reveal the encoded information. So a more realistic goal of cryptography is to make obtaining the information too work-intensive or time-consuming to be worthwhile to the attacker.

The first encryption methods date back to 4000 years ago and were considered more of an art form. Encryption was later adapted as a tool to use in warfare, commerce, government, and other arenas in which secrets needed to be safeguarded. With the relatively recent birth of the Internet, encryption has gained new prominence as a vital tool in everyday transactions. Throughout history, individuals and governments have worked to protect communication by encrypting it. As a result, the encryption algorithms and the devices that use them have increased in complexity, new methods and algorithms have been continually introduced, and encryption has become an integrated part of the computing world.

Cryptography has had an interesting history and has undergone many changes down through the centuries. Keeping secrets has proven very important to the workings of civilization. It gives individuals and groups the ability to hide their true intentions, gain a competitive edge, and reduce vulnerability, among other things.

The changes that cryptography has undergone closely follow advances in technology. The earliest cryptography methods involved a person carving messages into wood or stone, which was then delivered to the intended individual who had the necessary means to decipher the messages. Cryptography has come a long way since then. Now it is inserted into streams of binary code that pass over network wires, Internet communication paths, and airwaves.

The History of Cryptography

Look, I scrambled up the message so no one can read it.
Response: Yes, but now neither can we.

Cryptography has roots that begin around 2000 B.C. in Egypt, when hieroglyphics were used to decorate tombs to tell the life story of the deceased. The intention of the practice was not so much about hiding the messages themselves; rather, the hieroglyphics were intended to make the life story seem more noble, ceremonial, and majestic.

Encryption methods evolved from being mainly for show into practical applications used to hide information from others.

A Hebrew cryptographic method required the alphabet to be flipped so each letter in the original alphabet was mapped to a different letter in the flipped, or shifted, alphabet. The encryption method was called *atbash*, which was used to hide the true meaning of messages. An example of an encryption key used in the atbash encryption scheme is shown next:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

For example, the word "security" is encrypted into "hvxfirgb." What does "xrhhk" come out to be?

This is an example of a ***substitution cipher,*** because each character is replaced with another character. This type of substitution cipher is referred to as a ***monoalphabetic substitution cipher*** because it uses only one alphabet, whereas a ***polyalphabetic substitution cipher*** uses multiple alphabets.

NOTE Cipher is another term for algorithm.

This simplistic encryption method worked for its time and for particular cultures, but eventually more complex mechanisms were required.

Around 400 B.C., the Spartans used a system of encrypting information in which they would write a message on a sheet of papyrus (a type of paper) that was wrapped around a staff (a stick or wooden rod), which was then delivered and wrapped around a different staff by the recipient. The message was only readable if it was wrapped around the correct size staff, which made the letters properly match up, as shown in Figure 8-1. This is referred to as the ***scytale cipher.*** When the papyrus was not wrapped around the staff, the writing appeared as just a bunch of random characters.

Later, in Rome, Julius Caesar (100–44 B.C.) developed a simple method of shifting letters of the alphabet, similar to the atbash scheme. He simply shifted the alphabet by three positions. The following example shows a standard alphabet and a shifted alphabet. The alphabet serves as the algorithm, and the key is the number of locations it has been shifted during the encryption and decryption process.

Standard Alphabet:
ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cryptographic Alphabet:
DEFGHIJKLMNOPQRSTUVWXYZABC

As an example, suppose we need to encrypt the message "Logical Security." We take the first letter of this message, *L,* and shift up three locations within the alphabet. The encrypted version of this first letter is *O,* so we write that down. The next letter to be encrypted is *O,* which matches *R* when we shift three spaces. We continue this process for the whole message. Once the message is encrypted, a carrier takes the encrypted version to the destination, where the process is reversed.

Plaintext:
LOGICAL SECURITY

Ciphertext:
ORJLFDO VHFXULWB

Today, this technique seems too simplistic to be effective, but in the time of Julius Caesar, not very many people could read in the first place, so it provided a high level of protection. The Caesar cipher is an example of a monoalphabetic cipher. Once more people could read and reverse-engineer this type of encryption process, the cryptographers of that day increased the complexity by creating polyalphabetic ciphers.

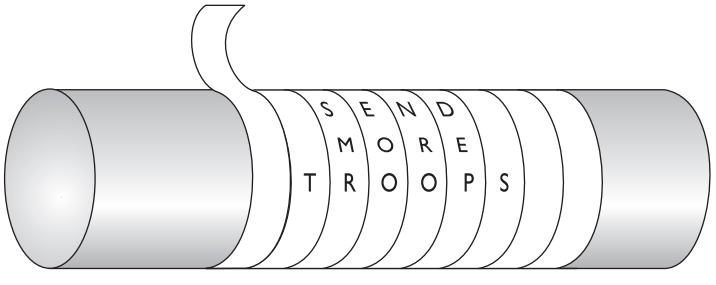


Figure 8-1
The scytale was used by the Spartans to decipher encrypted messages.

ROT13

A more recent encryption method used in the 1980s, ***ROT13***, was really the same thing as a Caesar cipher. Instead of shifting three spaces in the alphabet, the encryption process shifted 13 spaces. It was not really used to protect data, because our society could already easily handle this task. Instead, it was used in online forums (or bulletin boards) when "inappropriate" material, as in nasty jokes, were shared among users. The idea was that if you were interested in reading something potentially "offensive" you could simple use the shift 13 approach and read the material. Other people who did not want to view it would not be offended, because they would just leave the text and not decrypt it.

In the 16th century in France, Blaise de Vigenere developed a polyalphabetic substitution cipher for Henry III. This was based on the Caesar cipher, but it increased the difficulty of the encryption and decryption process.

As shown in Figure 8-2, we have a message that needs to be encrypted, which is SYSTEM SECURITY AND CONTROL. We have a key with the value of SECURITY. We also have a Vigenere table, or algorithm, which is really the Caesar cipher on steroids. Whereas the Caesar cipher used one shift alphabet (letters were shifted up three places), the Vigenere cipher has 27 shift alphabets and the letters are shifted up only one place.

NOTE Plaintext is the readable version of a message. After an encryption process, the resulting text is referred to as ciphertext.

So, looking at the example in Figure 8-2, we take the first value of the key, *S*, and, starting with the first alphabet in our algorithm, trace over to the *S* column. Then we look at the first value of plaintext that needs to be encrypted, which is *S*, and go down to the *S* row. We follow the column and row and see that they intersect on the value *K*. That is the first encrypted value of our message, so we write down *K*. Then we go to the next value in our key, which is *E*, and the next value of plaintext, which is *Y*. We see that the *E* column and the *Y* row intersect at the cell with the value of *C*. This is our second encrypted value, so we write that down. We continue this process for the whole message (notice that the key repeats itself, since the message is longer than the key). The resulting ciphertext is the encrypted form that is sent to the destination. The destination must have the same algorithm (Vigenere table) and the same key (SECURITY) to properly reverse the process to obtain a meaningful message.

The evolution of cryptography continued as countries refined their practices using new methods, tools, and practices throughout the Middle Ages. By the late 1800s, cryptography was commonly used in the methods of communication between military factions.

During World War II, encryption devices were used for tactical communication, which drastically improved with the mechanical and electromechanical technology that provided the world with telegraphic and radio communication. The rotor cipher machine, which is a device that substitutes letters using different rotors within the machine, was a huge breakthrough in military cryptography that provided complexity that proved difficult to break. This work gave way to the most famous cipher machine in

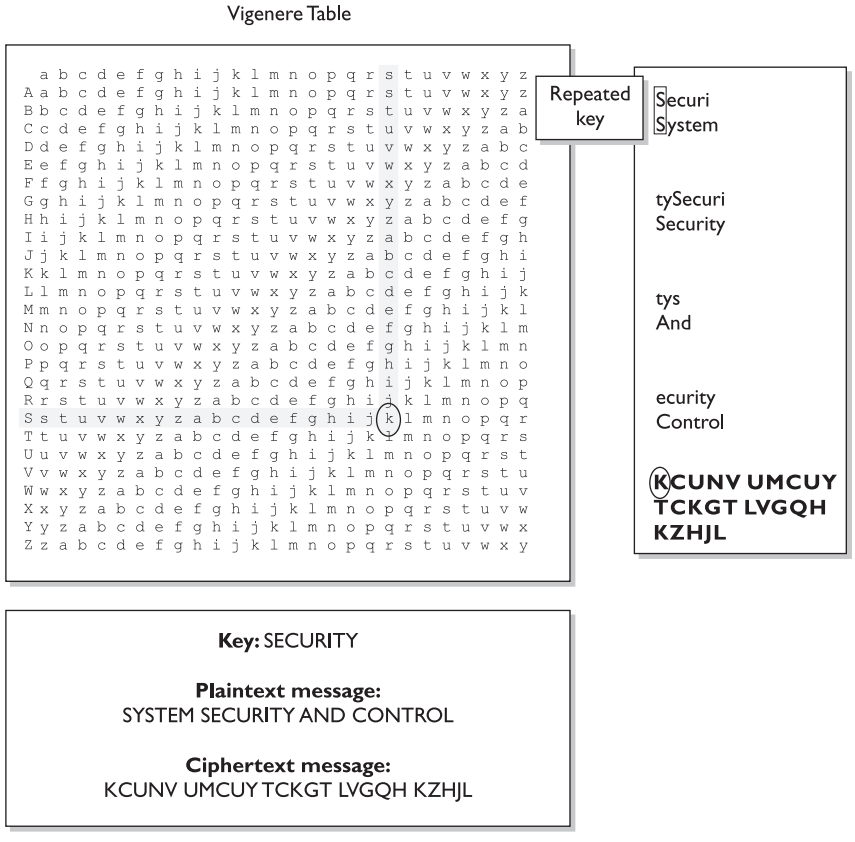


Figure 8-2 Polyalphabetic algorithms were developed to increase encryption complexity.

history to date: Germany's ***Enigma*** machine. The Enigma machine had separate rotors, a plugboard, and a reflecting rotor.

The originator of the message would configure the Enigma machine to its initial settings before starting the encryption process. The operator would type in the first letter of the message, and the machine would substitute the letter with a different letter and present it to the operator. This encryption was done by moving the rotors a predefined number of times. So, if the operator typed in a *T* as the first character, the Enigma machine might present an *M* as the substitution value. The operator would write down the letter *M* on his sheet. The operator would then advance the rotors and enter the next letter. Each time a new letter was to be encrypted, the operator would advance the rotors to a new setting. This process was followed until the whole message was encrypted. Then the encrypted text was transmitted over the airwaves, most likely to a German U-boat. The chosen substitution for each letter was dependent upon the rotor setting, so the crucial and secret part of this process (the key) was the initial setting and how the operators advanced the rotors when encrypting and decrypting a message. The operators at each end needed to know this sequence of increments to advance each rotor in order to enable the German military units to properly communicate.

Although the mechanisms of the Enigma were complicated for the time, a team of Polish cryptographers broke its code and gave Britain insight into Germany's attack plans and military movement. It is said that breaking this encryption mechanism shortened World War II by two years. After the war, details about the Enigma machine were published—one of the machines is exhibited at the Smithsonian Institute.

Cryptography has a deep, rich history. Mary, Queen of Scots, lost her life in the 16th century when an encrypted message she sent was intercepted. During the Revolutionary War, Benedict Arnold used a codebook cipher to exchange information on troop movement and strategic military advancements. Militaries have always played a leading role in using cryptography to encode information and to attempt to decrypt the enemy's encrypted information. William Frederick Friedman, who published *The Index of Coincidence and Its Applications in Cryptography* in 1920, is called the "Father of Modern Cryptography" and broke many messages intercepted during WWII. Encryption has been used by many governments and militaries and has contributed to great victory for some because it enabled them to execute covert maneuvers in secrecy. It has also contributed to great defeat for others, when their cryptosystems were discovered and deciphered.

When computers were invented, the possibilities for encryption methods and devices expanded exponentially and cryptography efforts increased dramatically. This era brought unprecedented opportunity for cryptographic designers to develop new encryption techniques. The most well-known and successful project was ***Lucifer,*** which was developed at IBM. Lucifer introduced complex mathematical equations and functions that were later adopted and modified by the U.S. National Security Agency (NSA) to establish the U.S. Data Encryption Standard (DES) in 1976, a federal government standard. DES has been used worldwide for financial and other transactions, and was imbedded into numerous commercial applications. DES has had a rich history in computer-oriented encryption and has been in use for over 25 years.

A majority of the protocols developed at the dawn of the computing age have been upgraded to include cryptography and to add necessary layers of protection. Encryption is used in hardware devices and in software to protect data, banking transactions, corporate extranet transmissions, e-mail messages, web transactions, wireless communications, the storage of confidential information, faxes, and phone calls.

The code breakers and cryptanalysis efforts and the amazing number-crunching capabilities of the microprocessors hitting the market each year have quickened the evolution of cryptography. As the bad guys get smarter and more resourceful, the good guys must increase their efforts and strategy. ***Cryptanalysis*** is the science of studying and breaking the secrecy of encryption processes, compromising authentication schemes, and reverse-engineering algorithms and keys. Cryptanalysis is an important piece of cryptography and cryptology. When carried out by the "good guys," cryptanalysis is intended to identify flaws and weaknesses so developers can go back to the drawing board and improve the components. It is also performed by curious and motivated hackers, to identify the same types of flaws, but with the goal of obtaining the encryption key for unauthorized access to confidential information.

NOTE Cryptanalysis is a very sophisticated science that encompasses a wide variety of tests and attacks. We will cover these types of attacks at the end of this chapter. Cryptology, on the other hand, is the study of cryptanalysis and cryptography.

Different types of cryptography have been used throughout civilization, but today cryptography is deeply rooted in every part of our communications and computing world. Automated information systems and cryptography play a huge role in the effectiveness of militaries, the functionality of governments, and the economics of private businesses. As our dependency upon technology increases, so does our dependency upon cryptography, because secrets will always need to be kept.

References

- **"A Short History of Cryptography," by Shon Harris, *Information Security Magazine* (July 2001)** www.infosecuritymag.com/articles/july01/columns_logoff.shtml
- **Chapter 2.1, "Security Strategies for E-Companies," by Fred Cohen** http://all.net/books/ip/Chap2-1.html
- **"An Introduction to Cryptography"** http://home.earthlink.net/~mylnir/crypt.intro.html
- **Trinity College Department of Computer Science Historical Cryptography web site** http://starbase.trincoll.edu/~crypto
- **Open Directory Project Historical Cryptography links** http://dmoz.org/Science/Math/Applications/Communication_Theory/Cryptography/Historical

Cryptography Definitions and Concepts

Why can't I read this?
Response: It is in ciphertext.

Encryption is a method of transforming readable data, called ***plaintext,*** into a form that appears to be random and unreadable, which is called ***ciphertext.*** Plaintext is in a form that can be understood either by a person (a document) or by a computer (executable code). Once it is transformed into ciphertext, neither human nor machine can properly process it until it is decrypted. This enables the transmission of confidential information over insecure channels without unauthorized disclosure. When data are stored on a computer, they are usually protected by logical and physical access controls. When this same sensitive information is sent over a network, it can no longer take these controls for granted, and the information is in a much more vulnerable state.

A system or product that provides encryption and decryption is referred to as a ***cryptosystem*** and can be created through hardware components or program code in an application. The cryptosystem uses an encryption algorithm (which determines how simple or complex the encryption process will be), keys, and the necessary software components and protocols. Most algorithms are complex mathematical formulas that are applied in a specific sequence to the plaintext. Most encryption methods use a secret value called a key (usually a long string of bits), which works with the algorithm to encrypt and decrypt the text.

The ***algorithm,*** the set of rules, dictates how enciphering and deciphering take place. Many of the mathematical algorithms used in computer systems today are publicly known and are not the secret part of the encryption process. If the internal mechanisms of the algorithm are not a secret, then something must be. The secret piece of using a well-known encryption algorithm is the key. A common analogy used to illustrate this point is the use of locks you would purchase from your local hardware store. Let's say 20 people bought the same brand of lock. Just because these people share the same type and brand of lock does not mean they can now unlock each other's doors and gain access to their private possessions. Instead, each lock comes with its own key, and that one key can only open that one specific lock.

In encryption, the ***key*** (cryptovariable) is a value that comprises a large sequence of random bits. Is it just any random number of bits crammed together? Not really. An algorithm contains a ***keyspace,*** which is a range of values that can be used to construct a key. When the algorithm needs to generate a new key, it uses random values from this keyspace. The larger the keyspace, the more available values can be used to represent different keys—and the more random the keys are, the harder it is for intruders to figure them out. For example, if an algorithm allows a key length of 2 bits, the keyspace for that algorithm would be 4, which indicates the total number of different keys that would be possible. (Remember that we are working in binary and that 2^2 equals 4.) That would not be a very large keyspace, and certainly it would not take an attacker very long to find the correct key that was used.

A large keyspace allows for more possible keys. (Today, we are commonly using key sizes of 128, 256, or 512 bits. So a key size of 512 bits would provide a 2^{512} keyspace.) The encryption algorithm should use the entire keyspace and choose the values to make up the keys as randomly as possible. If a smaller keyspace were used, there would be fewer values to choose from when generating a key, as shown in Figure 8-3. This would increase an attacker's chance of figuring out the key value and deciphering the protected information.

If an eavesdropper captures a message as it passes between two people, she can view the message, but it appears in its encrypted form and is therefore unusable. Even if this attacker knows the algorithm that the two people are using to encrypt and decrypt their information, without the key, this information remains useless to the eavesdropper, as shown in Figure 8-4.

Cryptosystems

A cryptosystem encompasses all of the necessary components for encryption and decryption to take place. Pretty Good Privacy (PGP) is just one example of a cryptosystem. A cryptosystem is made up of at least the following:

- Software
- Protocols
- Algorithms
- Keys

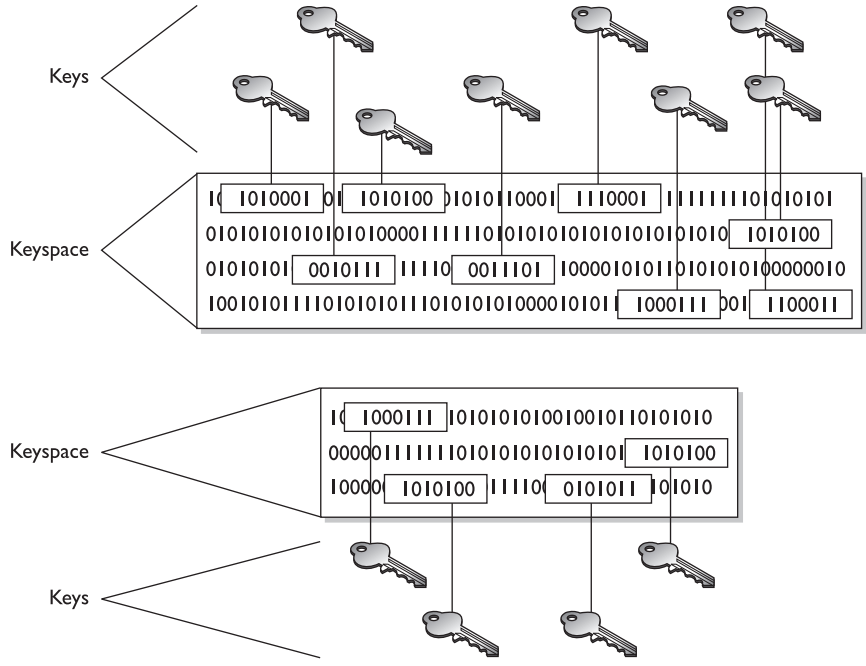


Figure 8-3 Larger keyspaces permit a greater number of possible key values.

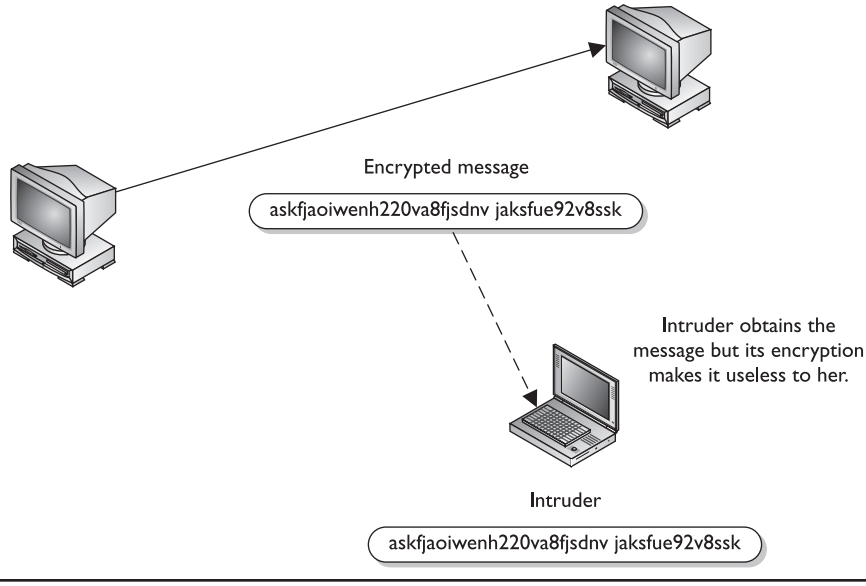


Figure 8-4 Without the right key, the captured message is useless to an attacker.

Kerckhoffs' Principle

Auguste Kerckhoffs published a paper in 1883 stating that the only secrecy involved with a cryptography system should be the key. He claimed that the algorithm should be publicly known. He asserted that if security were based on too many secrets, there would be more vulnerabilities to possibly exploit.

So, why do we care what some guy said over 120 years ago? Because this debate is still going on. Cryptographers in the private and academic sectors agree with Kerckhoffs' principle, because making an algorithm publicly available means that many more people can view the source code, test it, and uncover any type of flaws or weaknesses. It is the attitude of "many heads are better than one." Once someone uncovers some type of flaw, the developer can fix the issue and provide society with a much stronger algorithm.

But, not everyone agrees with this philosophy. Governments around the world create their own algorithms that are not released to the public. Their stance is that if a smaller number of people know how the algorithm actually works, then a smaller number of people will know how to possibly break it. Cryptographers in the private sector do not agree with this practice and do not trust algorithms they cannot examine.

It is basically the same as the open-source versus compiled software debate that is in full force today.

The Strength of the Cryptosystem

You are the weakest link. Goodbye!

The ***strength*** of an encryption method comes from the algorithm, the secrecy of the key, the length of the key, the initialization vectors, and how they all work together within the cryptosystem. When strength is discussed in encryption, it refers to how hard it is to figure out the algorithm or key, whichever is not made public. Attempts to break a cryptosystem usually involve processing an amazing number of possible values in the hopes of finding the one value (key) that can be used to decrypt a specific message. The strength of an encryption method correlates to the amount of necessary processing power, resources, and time required to break the cryptosystem or figure out the value of the key. Breaking a cryptosystem can be accomplished by a brute force attack, which means trying every possible key value until the resulting plaintext is meaningful. Depending on the algorithm and length of the key, this can be an easy task or one that is close to impossible. If a key can be broken with a Pentium II processor in three hours, the cipher is not strong at all. If the key can only be broken with the use of a thousand multiprocessing systems over 1.2 million years, then it is pretty darn strong.

NOTE Initialization vectors are explained in the section with the same name later in this chapter.

The goal when designing an encryption method is to make compromising it too expensive or too time-consuming. Another name for cryptography strength is ***work factor,*** which is an estimate of the effort and resources it would take an attacker to penetrate a cryptosystem.

How strong of a protection mechanism is required depends on the sensitivity of the data being protected. It is not necessary to encrypt information about a friend's Saturday barbeque with a top-secret encryption algorithm. Conversely, it is not a good idea to send intercepted spy information using PGP. Each type of encryption mechanism has its place and purpose.

Even if the algorithm is very complex and thorough, other issues within encryption can weaken encryption methods. Because the key is usually the secret value needed to actually encrypt and decrypt messages, improper protection of the key can weaken the encryption. Even if a user employs an algorithm that has all the requirements for strong encryption, including a large keyspace and a large and random key value, if she shares her key with others, the strength of the algorithm becomes almost irrelevant.

Important elements of encryption are to use an algorithm without flaws, use a large key size, use all possible values within the keyspace, and protect the actual key. If one element is weak, it could be the link that dooms the whole process.

Services of Cryptosystems

Cryptosystems can provide the following services:

- **Confidentiality** Render the information unintelligible except by authorized entities
- **Integrity** Data has not been altered in an unauthorized manner since it was created, transmitted, or stored
- **Authentication** Verify the identity of the user or system that created information
- **Authorization** Upon proving identity, the individual is then provided with the key or password that will allow access to some resource
- **Nonrepudiation** Ensures that the sender cannot deny sending the message

As an example of how these services work, suppose your boss sends you a message telling you that you will be receiving a raise that doubles your salary. The message is encrypted, so you can be sure it really came from your boss (authenticity), that someone did not alter it before it arrived at your computer (integrity), that no one else was able to read it as it traveled over the network (confidentiality), and that your boss cannot deny sending it later when he comes to his senses (nonrepudiation).

Different types of messages and transactions require higher or lower degrees of one or all of the services that cryptography methods can supply. Military and intelligence agencies are very concerned about keeping information confidential, so they would choose encryption mechanisms that provide a high degree of secrecy. Financial institutions care about confidentiality, but they also care about the integrity of the data being transmitted, so the encryption mechanism they would choose may differ from the military's encryption methods. If messages were accepted that had a misplaced decimal point or zero, the ramifications could be far reaching in the financial world. Legal agencies may care most about the authenticity of the messages they receive. If information

received ever needed to be presented in a court of law, its authenticity would certainly be questioned; therefore, the encryption method used must ensure authenticity, which confirms who sent the information.

NOTE If David sends a message and then later claims he did not send it, this is an act of repudiation. When a cryptography mechanism provides nonrepudiation, the sender cannot later deny they sent the message (well, they can try to deny it, but the cryptosystem proves otherwise). It's a way of keeping the sender honest.

The types and uses of cryptography have increased over the years. At one time, cryptography was mainly used to keep secrets secret (confidentiality), but today we use cryptography to ensure the integrity of data, to authenticate messages, to confirm that a message was received, for access control, and much more. Throughout this chapter, we will cover the different types of cryptography that provide these different types of functionality, along with any related security issues.

Cryptography Definitions

The following definitions are critical for your understanding of cryptography:

- **Access control** Restricting and controlling subject and object access attempts
- **Algorithm** Set of mathematical rules used in encryption and decryption
- **Cipher** Another name for algorithm
- **Cryptography** Science of secret writing that enables you to store and transmit data in a form that is available only to the intended individuals
- **Cryptosystem** Hardware or software implementation of cryptography that transforms a message to ciphertext and back to plaintext
- **Cryptanalysis** Practice of breaking cryptic systems
- **Cryptology** The study of both cryptography and cryptanalysis
- **Data origin authentication** Proving the source of a message (system-based authentication)
- **Encipher** Act of transforming data into an unreadable format
- **Entity authentication** Proving the identity of the entity that sent a message
- **Decipher** Act of transforming data into a readable format
- **Key** Secret sequence of bits and instructions that governs the act of encryption and decryption

- **Key clustering** Instance when two different keys generate the same ciphertext from the same plaintext
- **Keyspace** A range of possible values used to construct keys
- **Plaintext** Data in readable format, also referred to as cleartext
- **Receipt** Acknowledgment that a message has been received
- **Work factor** Estimated time, effort, and resources necessary to break a cryptosystem

If some of these terms do not make sense now, just hold on. We will cover them all in the following sections.

One-Time Pad

I want to use my one-time pad three times.
Response: Not a good idea.

A ***one-time pad*** is a perfect encryption scheme because it is considered unbreakable if implemented properly. It was invented by Gilbert Vernam in 1917, so sometimes it is referred to as the Vernam cipher.

This cipher does not use shift alphabets, as do the Caesar and Vigenere ciphers discussed earlier, but instead uses a pad made up of random values, as shown in Figure 8-5. Our plaintext message that needs to be encrypted has been converted into bits, and our one-time pad is made up of random bits. This encryption process uses a binary mathematic function called exclusive-OR, usually abbreviated as XOR.

XOR is an operation that is applied to two bits and is a function commonly used in binary mathematics and encryption methods. When combining the bits, if both values are the same, the result is 0 (1 XOR 1 = 0). If the bits are different from each other, the result is 1 (1 XOR 0 = 1). For example:

Message stream `1001010111`

Keystream `0011101010`

Ciphertext stream `1010111101`

So in our example, the first bit of the message is XORed to the first bit of the one-time pad, which results in the ciphertext value 1. The second bit of the message is XORed with the second bit of the pad, which results in the value 0. This process continues until the whole message is encrypted. The result is the encrypted message that is sent to the receiver.

In Figure 8-5, we also see that the receiver must have the same one-time pad to decrypt the message, by reversing the process. The receiver takes the first bit of the encrypted message and XORs it with the first bit of the pad. This results in the plaintext value. The receiver continues this process for the whole encrypted message, until the entire message is decrypted.

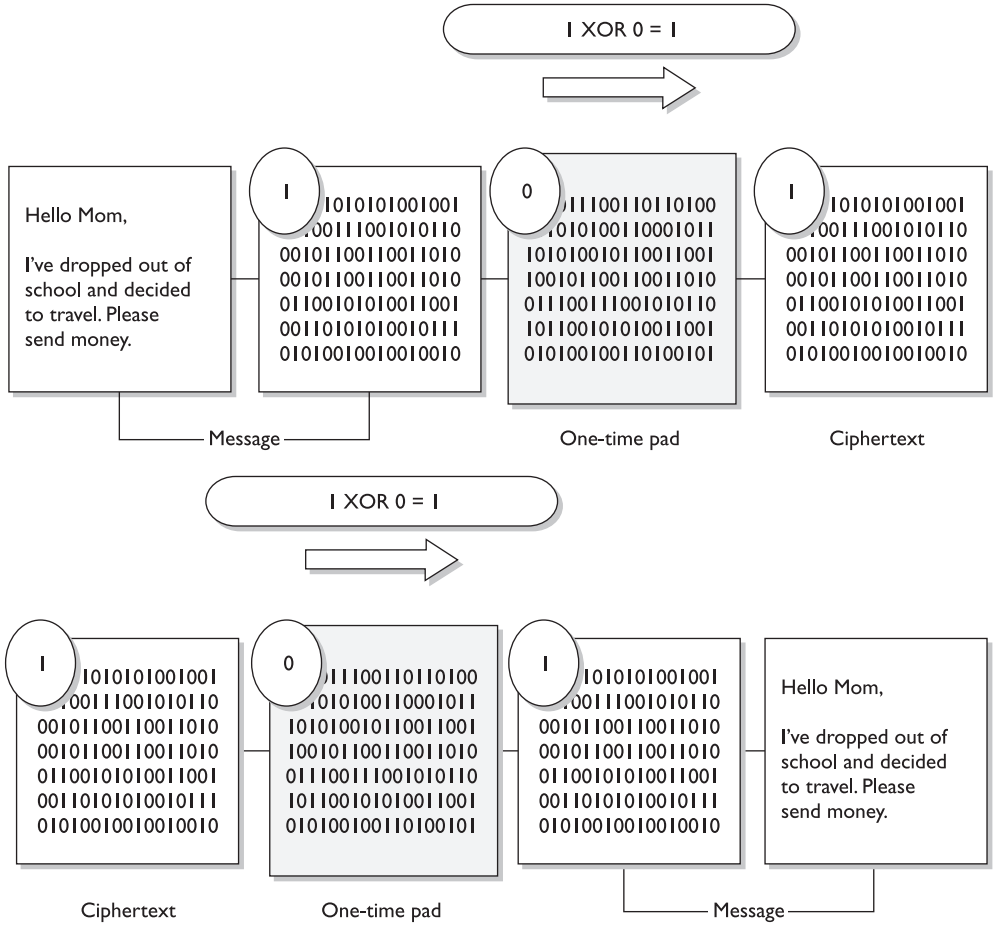


Figure 8-5 A one-time pad

The one-time pad encryption scheme is deemed unbreakable only if the following things are true about the implementation process:

- *The pad must be used only one time*. If the pad is used more than one time, this might introduce patterns in the encryption process that will aid the evildoer in his goal of breaking the encryption.
- *The pad must be as long as the message*. If it is not as long as the message, the pad will need to be reused to cover the whole message. This would be the same thing as using a pad more than one time, which could introduce patterns.
- *The pad must be securely distributed and protected at its destination*. This is a very cumbersome process to accomplish, because the pads are usually just individual pieces of paper that need to be delivered by a secure courier and properly guarded at each destination.

- *The pad must be made up of truly random values*. This may not seem like a difficult task, but even our computer systems today do not have truly random number generators; rather, they have pseudorandom number generators.

NOTE A number generator is used to create a stream of random values and must be seeded by an initial value. This piece of software obtains its seeding value from some component within the computer system (time, CPU cycles, and so on). Although a computer system is complex, it is a predictable environment, so if the seeding value is predictable in any way, the resulting values created are not truly random—but pseudorandom.

Although the one-time pad approach to encryption can provide a very high degree of security, it is impractical in most situations because of all of its different requirements. Each possible pair of entities that might want to communicate in this fashion must receive, in a secure fashion, a pad that is as long as, or longer than, the actual message. This type of key management can be overwhelming and may require more overhead than it is worth. The distribution of the pad can be challenging, and the sender and receiver must be perfectly synchronized so each is using the same pad.

One-time pads have been used throughout history to protect different types of sensitive data. Today, they are still in place for many types of militaries as a backup encryption option if current encryption processes (that require computers and a power source) are unavailable for reasons of war or attacks.

One-Time Pad Requirements

For a one-time pad encryption scheme to be considered unbreakable, each pad in the scheme must be:

- Made up of truly random values
- Used only one time
- Securely distributed to destination
- Secured at sender's and receiver's sites
- At least as long as the message

Running and Concealment Ciphers

I have my decoder ring, spyglasses, and secret handshake. Now let me figure out how I will encrypt my messages.

Two spy-novel-type ciphers are the running key cipher and the concealment cipher. The ***running key cipher*** could use a key that does not require an electronic algorithm and bit alterations, but cleverly uses components in the physical world around you. For instance, the algorithm could be a set of books agreed upon by the sender and receiver. The key in this type of cipher could be a book page, line number, and column count. If I get a message from my super-secret spy buddy and the message reads "149l6c7.299l3

c7.911l5c8," this could mean for me to look at the 1st book in our predetermined series of books, the 49th page, 6th line down the page, and the 7th column. So I write down the letter in that column, which is *m*. The second set of numbers starts with 2, so I go to the 2nd book, 99th page, 3rd line down, and then to the 7th column, which is *p*. The last letter I get from the 9th book, 11th page, 5th line, 8th column, which is *t*. So now I have come up with my important secret message, which is *mpt*. This means nothing to me, and I need to look for a new spy buddy. Running key ciphers can be used in different and more complex ways, but I think you get the point.

A ***concealment cipher*** is a message within a message. If my other super-secret spy buddy and I decide our key value is every third word, then when I get a message from him, I will pick out every third word and write it down. Suppose he sends me a message that reads, "The saying, 'The time is right' is not cow language, so is now a dead subject." Because my key is every third word, I come up with "The right cow is dead." This again means nothing to me, and I am now turning in my decoder ring.

No matter which of these two types of cipher is used, the roles of the algorithm and key are the same, even if they are not mathematical equations. In the running key cipher, the algorithm may be a predefined set of books. The key indicates the book, page, line, and word within that line. In substitution ciphers, the algorithm dictates that substitution will take place using a predefined alphabet or sequence of characters, and the key indicates that each character will be replaced with another character, as in the third character that follows it in that sequence of characters. In actual mathematical structures, the algorithm is a set of mathematical functions that will be performed on the message, and the key can indicate in which order these functions take place. So even if an attacker knows the algorithm, and we have to assume he does, if he does not know the key, the message is still useless to him.

Reference

- **Classical Cryptography** www.math.cudenver.edu/~wcherowi/courses/m5410/m5410cc.html

Steganography

Where's the top-secret message?
Response: In this picture of my dogs.

Steganography is a method of hiding data in another media type so the very existence of the data is concealed. Steganography is mainly accomplished by hiding messages in graphic images. The least significant bit of each byte of the image can be replaced with bits of the secret message. This practice does not affect the graphic enough to be detected.

Steganography does not use algorithms or keys to encrypt information. This is a process to hide data within another object so no one will detect its presence. A message can be hidden in a WAV file, in a graphic, or in unused spaces on a hard drive or sectors that are marked as unusable. Steganography can also be used to insert a digital watermark on digital images so illegal copies of the images can be detected.

References

- **Steganography and digital watermarking resource links, Johnson & Johnson Technology Consultants** www.jjtc.com/Steganography
- **"Steganography Revealed," by Kristy Westphal, SecurityFocus (April 9, 2003)** www.securityfocus.com/infocus/1684

Governmental Involvement in Cryptography

Big Brother is watching you! Um, I mean we are only watching the bad guys.

In the United States, in the 1960s to 1980s, exportation of cryptographic mechanisms and equipment was very carefully regulated and monitored. The goal was to make obtaining and using encryption technology harder for terrorists and criminals. Harry Truman created the NSA in 1952, and its main mission was, and still is, to listen in on communications in the interest of national security for the United States. The NSA keeps an extremely low profile, and its activities are highly secret. The NSA also conducts research in cryptology to create secure algorithms and to break other cryptosystems to enable eavesdropping and spying.

The government attempted to restrict the use of public cryptography so enemies of the United States could not employ encryption methods that were too strong for it to break. These steps caused tension and controversy between cryptography researchers, vendors, and the NSA pertaining to new cryptographic methods and the public use of them. The fear of those opposed to the restrictions was that if the government controlled all types of encryption and was allowed to listen in on private citizens' conversations, the obtained information would be misused in "Big Brotherly" ways. Also, if the government had the technology to listen in on everyone's conversations, the possibility existed that this technology would fall into the wrong hands, and be used for the wrong reasons.

At one time a group existed whose duty was to control the export of specific types of weapons and cryptographic products to communist countries. This group came up with the Coordinating Committee on Multilateral Export Controls (COCOM). Because the threat of communism decreased over time, this group was disbanded. Then, in 1996, a group of 33 countries reached an agreement to control exportation of the same types of items to several countries deemed to be "terrorist states." These countries (Iran, Iraq, Libya, North Korea, Sudan, Cuba, and Syria) were identified as having connections with terrorist groups and activities. The group set up agreed-upon guidelines regarding how to regulate exportation of certain types of weapons and technologies that contained cryptography functionality. In part, this group worked together to ensure "dual-use" products (products that have both civilian and military application) that contain encryption capabilities were not made available to the "terrorist states." Because one of the main goals of every military is to be able to eavesdrop on its perceived enemies, the group of 33 countries was concerned that if terrorist states were able to obtain strong encryption methods, spying on them would be much harder to accomplish.

Just as the United States has the NSA, different countries have government agencies that are responsible for snooping on the communications of potential enemies, which involves using very powerful systems that can break a certain level of encryption. Since these countries know, for example, that they can break encryption methods that use symmetric keys of up to 56 bits, they will allow these types of products to be exported in an uncontrolled manner. Anything using a symmetric key over 56 bits needs to be controlled, because the governments are not sure they can efficiently crack those codes.

The following outlines the characteristics of specific algorithm types that are considered too dangerous to fall into the hands of the enemy and thus are restricted:

- Symmetric algorithms with key sizes over 56 bits
- Asymmetric algorithms that carry out factorization of an integer with key sizes over 512 bits (such as RSA)
- Asymmetric algorithms that compute discrete logarithms in a field with key sizes over 512 bits (such as El Gamal)
- Asymmetric algorithms that compute discrete logarithms in a group (not in a field) with key sizes over 112 bits (such as ECC)

The Wassenaar Arrangement contains the agreed-upon guidelines that this group of countries came up with, but the decision of whether or not to follow the guidelines has been left up to the individual countries. The United States has relaxed its export controls over the years and today exportation can take place to any country, other than the previously listed "terrorist states," after a technical review. If the product is an open-source product, then a technical review is not required, but it is illegal to provide this type of product directly to identified terrorist groups and countries. Also, a technical review is not necessary for exportation of cryptography to foreign subsidiaries of U.S. firms.

Types of Ciphers

Symmetric encryption ciphers come in two basic types: substitution and transposition (permutation). The ***substitution cipher*** replaces bits, characters, or blocks of characters with different bits, characters, or blocks. The ***transposition cipher*** does not replace the

original text with different text, but rather moves the original values around. It rearranges the bits, characters, or blocks of characters to hide the original meaning.

Substitution Ciphers

Give me your A *and I will change it out for an* M. *Now, no one can read your message. Response: That will fool them.*

A substitution cipher uses a key to dictate how the substitution should be carried out. In the ***Caesar cipher,*** each letter is replaced with the letter three places beyond it in the alphabet. The algorithm is the alphabet and the key is the instruction "shift up three."

As a simple example, if George uses the Caesar cipher with the English alphabet to encrypt the important message "meow," the encrypted message would be "phrz." Substitution is used in today's symmetric algorithms, but it is extremely complex compared to this example, which is only meant to show you the concept of how a substitution cipher works in its most simplistic form.

Transposition Ciphers

In a transposition cipher, the values are scrambled, or put into a different order. The key determines the positions the values are moved to, as illustrated in Figure 8-6.

This is a simplistic example of a transposition cipher and only shows one way of performing transposition. When implemented with complex mathematical functions, transpositions can become quite sophisticated and difficult to break. Symmetric algorithms employed today use both long sequences of complicated substitutions and transpositions on messages. The algorithm contains the possible ways that substitution and transposition processes *can* take place (represented in mathematical formulas). The key is used as the instructions for the algorithm, dictating exactly how these processes *will* happen and in what order. To understand the relationship between an algorithm and a key, let's look at

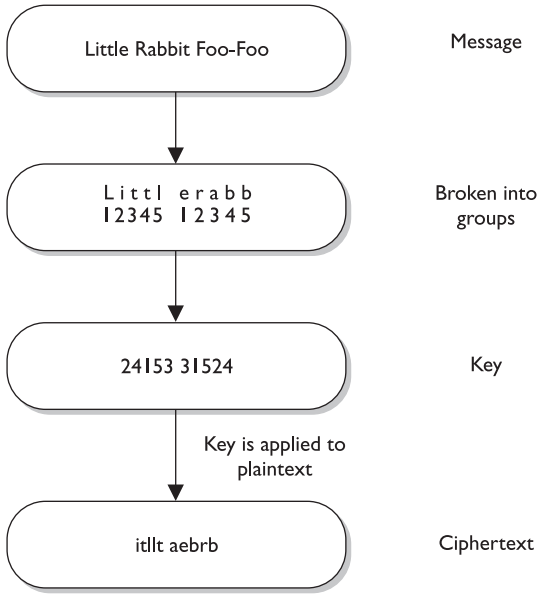


Figure 8-6
A transposition cipher

Figure 8-7. Conceptually, an algorithm is made up of different boxes, each of which has a different set of mathematical formulas that dictates the substitution and transposition steps that will take place on the bits that enter the box. To encrypt our message, the bit values must go through these different boxes. If each of our messages goes through each of these different boxes in the same order with the same values, the evildoer will be able to easily reverse-engineer this process and uncover our plaintext message.

To foil an evildoer, we use a key, which is a set of values that indicates which box should be used, in what order, and with what values. So if message A is encrypted with key 1, the key will make the message go through boxes 1, 6, 4, and then 5. When we need to encrypt message B, we will use key 2, which will make the message go through boxes 8, 3, 2, and then 9. It is the key that adds the randomness and the secrecy to the encryption process.

Simple substitution and transposition ciphers are vulnerable to attacks that perform ***frequency analysis.*** In every language, some words and patterns are used more often than others. For instance, in the English language, the most commonly used letter is *E*. If Mike is carrying out frequency analysis on a message, he will look for the most frequently repeated pattern of eight bits (which make up a character). So, if Mike sees that there are 12 patterns of eight bits and he knows that *E* is the most commonly used letter in the language, he will replace these bits with this vowel. This allows him to gain a foothold on the process, which will allow him to reverse-engineer the rest of the message.

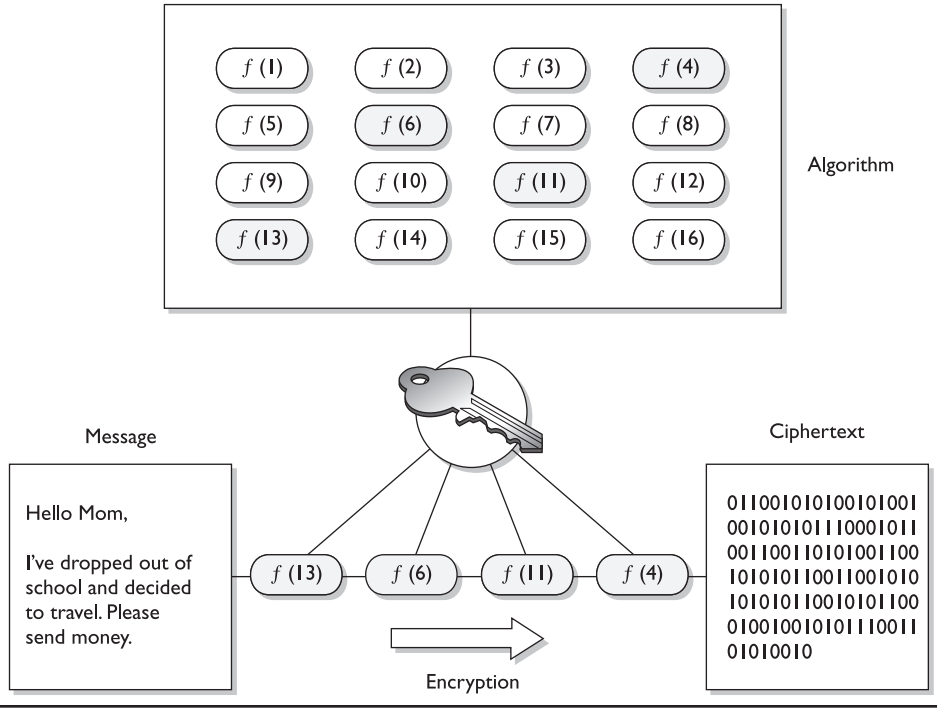


Figure 8-7 The algorithm and key relationship

Today's symmetric algorithms use substitution and transposition methods in their encryption processes, but the mathematics used are (or should be) too complex to allow for simplistic frequency-analysis attacks to be successful.

Methods of Encryption

Although there can be several pieces to an encryption process, the two main pieces are the algorithms and the keys. As stated earlier, algorithms used in computer systems are complex mathematical formulas that dictate the rules of how the plaintext will be turned into ciphertext. A key is a string of random bits that will be used by the algorithm to add to the randomness of the encryption process. For two entities to be able to communicate via encryption, they must use the same algorithm and, many times, the same key. In some encryption technologies, the receiver and the sender use the same key, and in other encryption technologies, they must use different but related keys for encryption and decryption purposes. The following sections explain the differences between these two types of encryption methods.

Symmetric vs. Asymmetric Algorithms

Cryptography algorithms are either ***symmetric algorithms,*** which use symmetric keys (also called secret keys), or ***asymmetric algorithms,*** which use asymmetric keys (also called public and private keys). As if encryption were not complicated enough, the terms used to describe the key types only make it worse. Just pay close attention and you will get through this fine.

Symmetric Cryptography

In a cryptosystem that uses symmetric cryptography, the sender and receiver use two instances of the same key for encryption and decryption, as shown in Figure 8-8. So the key has dual functionality, in that it can carry out both encryption and decryption processes. Symmetric keys are also called secret keys, because this type of encryption relies on each user to keep the key a secret and properly protected. If an intruder were to get this key, they could decrypt any intercepted message encrypted with it.

Each pair of users who want to exchange data using symmetric key encryption must have two instances of the same key. This means that if Dan and Iqqi want to communicate, both need to obtain a copy of the same key. If Dan also wants to communicate using symmetric encryption with Norm and Dave, he needs to have three separate keys, one for each friend. This might not sound like a big deal until Dan realizes that he may communicate with hundreds of people over a period of several months, and keeping track and using the correct key that corresponds to each specific receiver can become a daunting task. If ten people needed to communicate securely with each other using symmetric keys, then 45 keys would need to be kept track of. If 100 people were going to communicate, then 4950 keys would be involved. The equation used to calculate the number of symmetric keys needed is

$N(N - 1)/2$ = number of keys

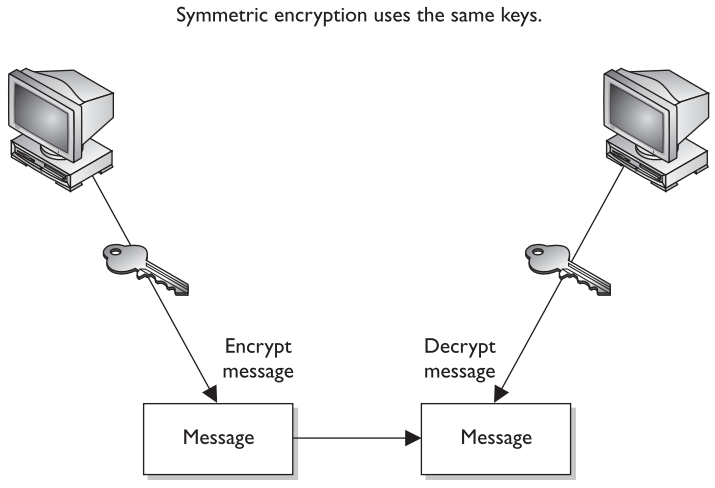


Figure 8-8 When using symmetric algorithms, the sender and receiver use the same key for encryption and decryption functions.

The security of the symmetric encryption method is completely dependent on how well users protect the key. This should raise red flags for you if you have ever had to depend on a whole staff of people to keep a secret. If a key is compromised, then all messages encrypted with that key can be decrypted and read by an intruder. This is complicated further by how symmetric keys are actually shared and updated when necessary. If Dan wants to communicate with Norm for the first time, Dan has to figure out how to get the right key to Norm securely. It is not safe to just send it in an e-mail message, because the key is not protected and can be easily intercepted and used by attackers. Thus, Dan must get the key to Norm through an ***out-of-band method.*** Dan can save the key on a thumb drive and walk over to Norm's desk, or have a secure courier deliver it to Norm. This is a huge hassle, and each method is very clumsy and insecure.

Because both users employ the same key to encrypt and decrypt messages, symmetric cryptosystems can provide confidentiality but they cannot provide authentication or nonrepudiation. There is no way to prove through cryptography who actually sent a message if two people are using the same key.

If symmetric cryptosystems have so many problems and flaws, why use them at all? Because they are very fast and can be hard to break. Compared with asymmetric systems, symmetric algorithms scream in speed. They can encrypt and decrypt relatively quickly large amounts of data that would take an unacceptable amount of time to encrypt and decrypt with an asymmetric algorithm. It is also difficult to uncover data encrypted with a symmetric algorithm if a large key size is used. For many of our applications that require encryption, symmetric key cryptography is the only option.

The following list outlines the strengths and weakness of symmetric key systems:

Strengths

- Much faster than asymmetric systems.
- Hard to break if using a large key size.

Weaknesses

- Requires a secure mechanism to deliver keys properly.
- Each pair of users needs a unique key, so as the number of individuals increases, so does the number of keys, possibly making key management overwhelming.
- Provides confidentiality but not authenticity or nonrepudiation.

The following are examples of symmetric algorithms, which will be explained later in the "Block and Stream Ciphers" section:

- Data Encryption Standard (DES)
- Triple-DES (3DES)
- Blowfish
- IDEA
- RC4, RC5, and RC6
- Advanced Encryption Standard (AES)

References

- ***Security in Open Systems*, Node 208, "Symmetric Key Cryptography," by Paul Markovitz, NIST Special Publication 800-7 (July 1994)** http://csrc.nist.gov/publications/nistpubs/800-7/node208.html
- **Understanding the Public Key Cryptography** www.ibm.com/developerworks/ibm/library/it-sinn1

Asymmetric Cryptography

Some things you can tell the public, but some things you just want to keep private.

In symmetric key cryptography, a single secret key is used between entities, whereas in public key systems, each entity has different keys, or ***asymmetric keys.*** The two different asymmetric keys are mathematically related. If a message is encrypted by one key, the other key is required in order to decrypt the message.

In a public key system, the pair of keys is made up of one public key and one private key. The ***public key*** can be known to everyone, and the ***private key*** must be known and used only by the owner. Many times, public keys are listed in directories and databases of e-mail addresses so they are available to anyone who wants to use these keys to encrypt or decrypt data when communicating with a particular person. Figure 8-9 illustrates the use of the different keys.

The public and private keys of an asymmetric cryptosystem are mathematically related, but if someone gets another person's public key, she should not be able to figure out the corresponding private key. This means that if an evildoer gets a copy of Bob's public key, it does not mean she can employ some mathematical magic and find out Bob's private key. But if someone got Bob's private key, then there is big trouble—no one other than the owner should have access to a private key.

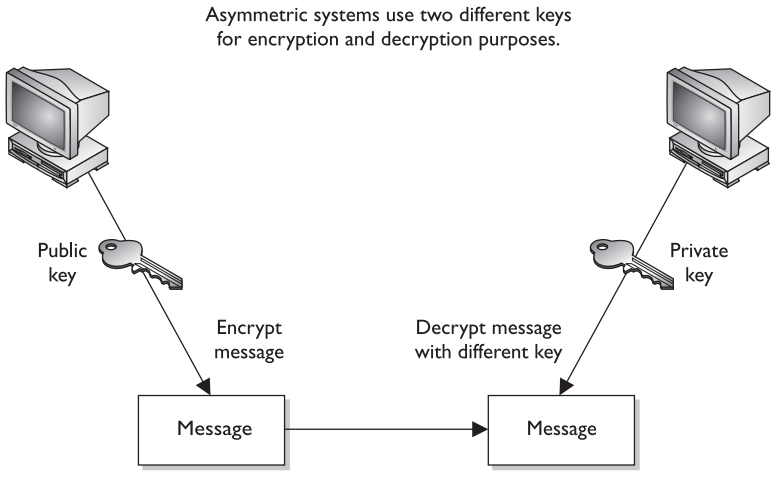


Figure 8-9
An asymmetric cryptosystem

If Bob encrypts data with his private key, the receiver must have a copy of Bob's public key to decrypt it. The receiver can decrypt Bob's message and decide to reply to Bob in an encrypted form. All she needs to do is encrypt her reply with Bob's public key, and then Bob can decrypt the message with his private key. It is not possible to encrypt and decrypt using the same key when using an asymmetric key encryption technology because, although mathematically related, the two keys are not the same key, as they are in symmetric cryptography. Bob can encrypt data with his private key, and the receiver can then decrypt it with Bob's public key. By decrypting the message with Bob's public key, the receiver can be sure the message really came from Bob. A message can be decrypted with a public key only if the message was encrypted with the corresponding private key. This provides authentication, because Bob is the only one who is supposed to have his private key. If the receiver wants to make sure Bob is the only one that can read her reply, she will encrypt the response with his public key. Only Bob will be able to decrypt the message because he is the only one who has the necessary private key.

The receiver can also choose to encrypt data with her private key instead of using Bob's public key. Why would she do that? Authentication—she wants Bob to know that the message came from her and no one else. If she encrypted the data with Bob's public key, it does not provide authenticity because anyone can get Bob's public key. If she uses her private key to encrypt the data, then Bob can be sure the message came from her and no one else. Symmetric keys do not provide authenticity because the same key is used on both ends. Using one of the secret keys does not ensure the message originated from a specific individual.

If confidentiality is the most important security service to a sender, she would encrypt the file with the receiver's public key. This is called a ***secure message format*** because it can only be decrypted by the person who has the corresponding private key.

If authentication is the most important security service to the sender, then she would encrypt the data with her private key. This provides assurance to the receiver that the only person who could have encrypted the data is the individual who has possession of that private key. If the sender encrypted the data with the receiver's public key, authentication is not provided because this public key is available to anyone.

Encrypting data with the sender's private key is called an ***open message format*** because anyone with a copy of the corresponding public key can decrypt the message. Confidentiality is not ensured.

Each key type can be used to encrypt and decrypt, so do not get confused and think the public key is only for encryption and the private key is only for decryption. They both have the capability to encrypt and decrypt data. However, if data are encrypted with a private key, they cannot be decrypted with a private key. If data are encrypted with a private key, they must be decrypted with the corresponding public key.

An asymmetric algorithm works much more slowly than a symmetric algorithm, because symmetric algorithms carry out relatively simplistic mathematical functions on the bits during the encryption and decryption processes. They substitute and scramble (transposition) bits, which is not overly difficult or processor-intensive. The reason it is hard to break this type of encryption is that the symmetric algorithms carry out this type of functionality over and over again. So a set of bits will go through a long series of being substituted and scrambled.

Asymmetric algorithms are slower than symmetric algorithms because they use much more complex mathematics to carry out their functions, which requires more processing time. Although they are slower, asymmetric algorithms can provide authentication and nonrepudiation, depending on the type of algorithm being used. Asymmetric systems also provide for easier and more manageable key distribution than symmetric systems and do not have the scalability issues of symmetric systems. The reason for these differences is that, with asymmetric systems, you can send out your public key to all of the people you need to communicate with, instead of keeping track of a unique key for each one of them. The "Hybrid Encryption Methods" section later in this chapter shows how these two systems can be used together to get the best of both worlds.

NOTE "Public key cryptography" is "asymmetric cryptography." The terms can be used interchangeably.

The following outlines the strengths and weaknesses of asymmetric key algorithms:

Strengths

- Better key distribution than symmetric systems
- Better scalability than symmetric systems
- Can provide authentication and nonrepudiation

Weaknesses

- Works much more slowly than symmetric systems
- Mathematically intensive tasks

The following are examples of asymmetric key algorithms:

- RSA
- Elliptic curve cryptosystem (ECC)
- Diffie-Hellman

- El Gamal
- Digital Signature Algorithm (DSA)
- Knapsack

These algorithms will be explained further in the "Types of Asymmetric Systems" section later in the chapter.

Table 8-1 summarizes the differences between symmetric and asymmetric algorithms.

References

- ***Security in Open Systems*, Node 210, "Asymmetric Key Cryptography," by Paul Markovitz, NIST Special Publication 800-7 (July 1994)** http://csrc.nist.gov/publications/nistpubs/800-7/node210.html
- **"Cryptography Defined/Brief History," by Sarah Simpson (Spring 1997)** www.eco.utexas.edu/~norman/BUS.FOR/course.mat/SSim/history.html
- **"Asymmetric Cryptography," by Daniel Steffen (March 1997)** www.maths.mq.edu.au/~steffen/old/PCry/report/node8.html
- **Frequently Asked Questions About Today's Cryptography, Version 4.1, Section 2.1.4.5, "What Is Output Feedback Mode?" by RSA Laboratories** www.rsasecurity.com/rsalabs/node.asp?id=2173
- **"Report on the Symmetric Key Block Cipher Modes of Operation Workshop, October 20, 2001"** http://csrc.nist.gov/CryptoToolkit/modes/workshop1/workshop-report.pdf

NOTE Digital signatures will be discussed later in the section "Digital Signatures." (We like to keep it simple around here.)

Attribute	Symmetric	Asymmetric
Keys	One key is shared between two or more entities.	One entity has a public key and the other entity has the corresponding private key.
Key exchange	Out-of-band through secure mechanisms.	A public key is made available to everyone and a private key is kept secret to the owner.
Speed	Algorithm is less complex and faster.	The algorithm is more complex and slower.
Use	Bulk encryption, which means encrypting files and communication paths.	Key distribution and digital signatures.
Security service provided	Confidentiality.	Authentication and nonrepudiation.

Table 8-1 Differences Between Symmetric and Asymmetric Systems

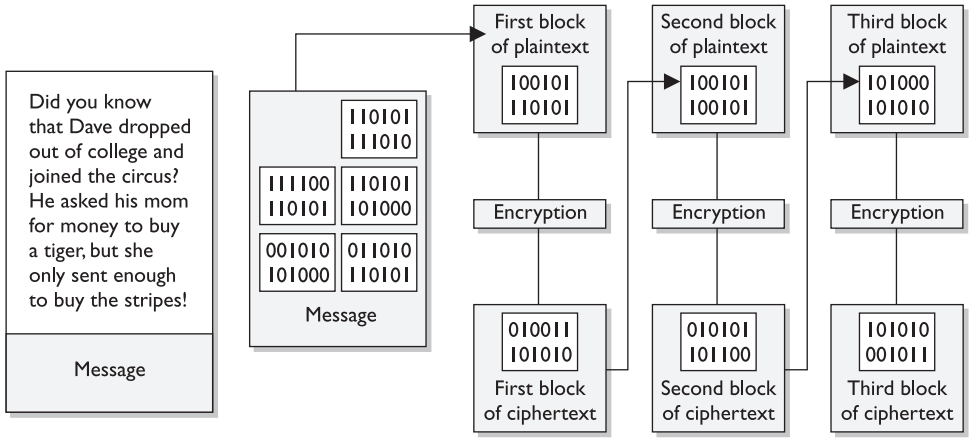
Block and Stream Ciphers

Which should I use, the stream cipher or the block cipher?
Response: The stream cipher, because it makes you look skinnier.

The two main types of symmetric algorithms are block ciphers, which work on blocks of bits, and stream ciphers, which work on one bit at a time.

Block Ciphers

When a ***block cipher*** is used for encryption and decryption purposes, the message is divided into blocks of bits. These blocks are then put through mathematical functions, one block at a time. Suppose you need to encrypt a message you are sending to your mother and you are using a block cipher that uses 64 bits. Your message of 640 bits is chopped up into 10 individual blocks of 64 bits. Each block is put through a succession of mathematical formulas, and what you end up with is 10 blocks of encrypted text. You send this encrypted message to your mother. She has to have the same block cipher and key, and those 10 ciphertext blocks go back through the algorithm in the reverse sequence and end up in your plaintext message.



A strong cipher contains the right level of two main attributes: confusion and diffusion. ***Confusion*** is commonly carried out through substitution, while ***diffusion*** is carried out by using transposition. For a cipher to be considered strong, it must contain both of these attributes, to ensure reverse-engineering is basically impossible. The randomness of the key values and the complexity of the mathematical functions dictate the level of confusion and diffusion involved.

In algorithms, diffusion takes place as individual bits of a block are scrambled, or diffused throughout that block. Confusion is provided by carrying out complex substitution functions so the bad guy cannot figure out how to substitute the right values and come up with the original plaintext. Suppose I have 500 wooden blocks with individual letters written on them. I line them all up to spell out a paragraph (plaintext). Then I substitute 300 of them with another set of 300 blocks (confusion through substitution). Then I scramble all of these blocks up (diffusion through transposition) and leave them in a pile. For you to figure out my original message, you would have to substitute the correct blocks and then put them back in the right order. Good luck.

Confusion pertains to making the relationship between the key and resulting ciphertext as complex as possible so the key cannot be uncovered from the ciphertext. Each ciphertext value should depend upon several parts of the key, but this mapping between the key values and the ciphertext values seems to be completely random to the observer.

Diffusion, on the other hand, means that a single plaintext bit has influence over several of the ciphertext bits. Changing a plaintext value should change many ciphertext values, not just one. In fact, in a strong block cipher, if one plaintext bit is changed, it will change every ciphertext bit with the probability of 50 percent. This means that if one plaintext bit changes, then about half of the ciphertext bits will change.

Block ciphers use diffusion and confusion in their methods. Figure 8-10 shows a conceptual example of a simplistic block cipher. It has four block inputs and each block is made up of four bits. The block algorithm has two layers of four-bit substitution boxes called *S-boxes*. Each S-box contains a lookup table used by the algorithm as instructions on how the bits should be encrypted.

Figure 8-10 shows that the key dictates what S-boxes are to be used when scrambling the original message from readable plaintext to encrypted nonreadable cipher-

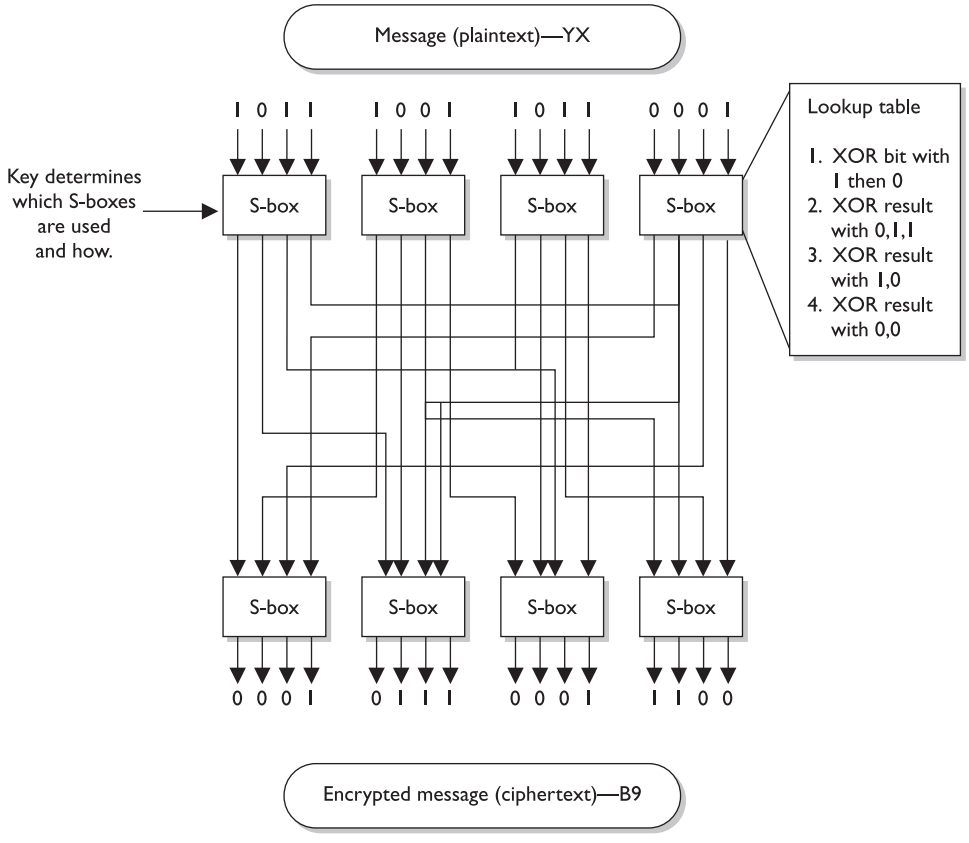


Figure 8-10 A message is divided into blocks of bits, and substitution and transposition functions are performed on those blocks.

text. Each S-box contains the different substitution and transposition methods that can be performed on each block. This example is simplistic—most block ciphers work with blocks of 32, 64, or 128 bits in size, and many more S-boxes are usually involved.

Stream Ciphers

As stated earlier, a block cipher performs mathematical functions on blocks of bits. A stream cipher, on the other hand, does not divide a message into blocks. Instead, a ***stream cipher*** treats the message as a stream of bits and performs mathematical functions on each bit individually.

When using a stream cipher, a plaintext bit will be transformed into a different ciphertext bit each time it is encrypted. Stream ciphers use ***keystream generators,*** which produces a stream of bits that is XORed with the plaintext bits to produce ciphertext, as shown in Figure 8-11.

NOTE This process is very similar to the one-time pad explained earlier. The individual bits in the one-time pad are used to encrypt the individual bits of the message through the XOR function, and in a stream algorithm the individual bits created by the keystream generator are used to encrypt the bits of the message through XOR also.

If the cryptosystem were only dependent upon the symmetric stream algorithm, an attacker could get a copy of the plaintext and the resulting ciphertext, XOR them together, and find the keystream to use in decrypting other messages. So the smart people decided to stick a key into the mix.

In block ciphers, it is the key that determines what functions are applied to the plaintext and in what order. The key provides the randomness of the encryption process. As stated earlier, most encryption algorithms are public, so people know how they work. The secret to the secret sauce is the key. In stream ciphers, the key also provides randomness, so that the stream of bits that are XORed to the plaintext are as random as possible. This concept is shown in Figure 8-12. As you can see in this graphic, both the sending and receiving ends must have the same key to generate the same keystream for proper encryption and decryption purposes.

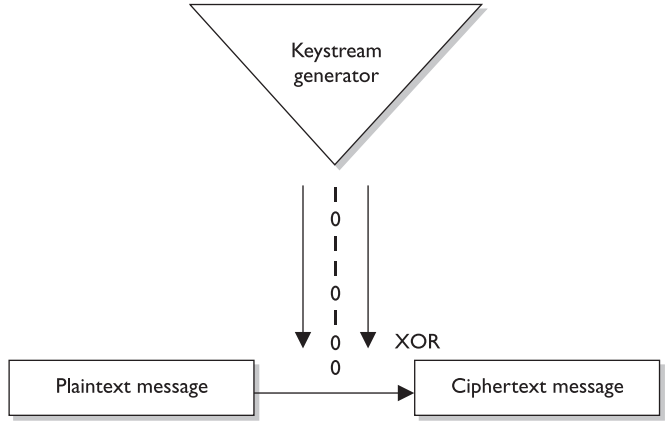


Figure 8-11 With stream ciphers, the bits generated by the keystream generator are XORed with the bits of the plaintext message.

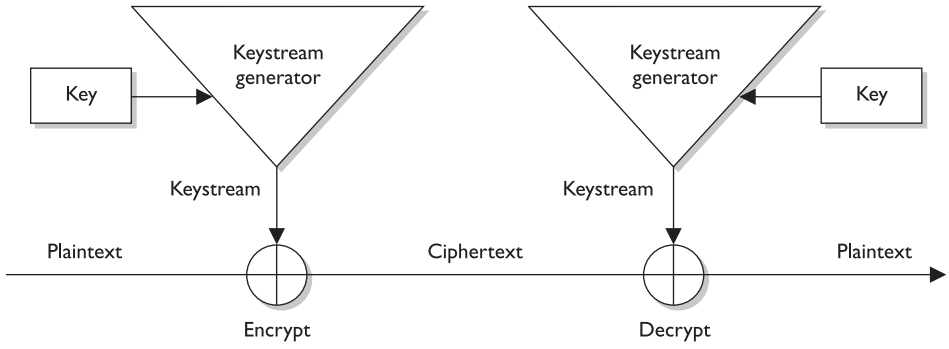


Figure 8-12 The sender and receiver must have the same key to generate the same keystream.

Initialization Vectors

Initialization vectors (IVs) are random values that are used with algorithms to ensure patterns are not created during the encryption process. They are used with keys and do not need to be encrypted when being sent to the destination. If IVs are not used, then two identical plaintext values that are encrypted with the same key will create the same ciphertext. Providing attackers with these types of patterns can make their job easier in breaking the encryption method and uncovering the key. For example, if we have the plaintext value of "See Spot run" two times within our message, we need to make sure that even though there is a pattern in the plaintext message, a pattern in the resulting ciphertext will not be created. So the IV and key are both used by the algorithm to provide more randomness to the encryption process.

A strong and effective stream cipher contains the following characteristics:

- **Long periods of no repeating patterns within keystream values** Bits generated by the keystream must be random.
- **Statistically unpredictable keystream** The bits generated from the keystream generator cannot be predicted.
- **A keystream not linearly related to the key** If someone figures out the keystream values, that does not mean she now knows the key value.
- **Statistically unbiased keystream (as many 0s as 1s)** There should be no dominance in the number of 0s or 1s in the keystream.

Stream ciphers require a lot of randomness and encrypt individual bits at a time. This requires more processing power than block ciphers require, which is why stream ciphers are better suited to be implemented at the hardware level. Because block ciphers do not require as much processing power, they can be easily implemented at the software level.

NOTE We do have block ciphers that work at the silicon level, and stream ciphers that work at the software level. The previous statement is just a "best practice" or guideline when it comes to development and implementation.

Stream Ciphers vs. One-Time Pads

Stream ciphers were developed to provide the same type of protection one-time pads do, which is why they work in such a similar manner. In reality, stream ciphers cannot provide the level of protection one-time pads do, but because stream ciphers are implemented through software and automated means, they are much more practical.

Hybrid Encryption Methods

Up to this point, we have figured out that symmetric algorithms are fast but have some drawbacks (lack of scalability, difficult key management, and they provide only confidentiality). Asymmetric algorithms do not have these drawbacks but are very slow. We just can't seem to win. So we turn to a hybrid system that uses symmetric and asymmetric encryption methods together.

Asymmetric and Symmetric Algorithms Used Together

Public key cryptography uses two keys (public and private) generated by an asymmetric algorithm for protecting encryption keys and key distribution, and a secret key is generated by a symmetric algorithm and used for bulk encryption. Then there is a hybrid use of the two different algorithms: asymmetric and symmetric. Each algorithm has its pros and cons, so using them together can be the best of both worlds.

In the hybrid approach, the two technologies are used in a complementary manner, with each performing a different function. A symmetric algorithm creates keys used for encrypting bulk data, and an asymmetric algorithm creates keys used for automated key distribution.

When a symmetric key is used for bulk data encryption, this key is used to encrypt the message you want to send. When your friend gets the message you encrypted, you want him to be able to decrypt it, so you need to send him the necessary symmetric key to use to decrypt the message. You do not want this key to travel unprotected, because if the message were intercepted and the key were not protected, an evildoer could intercept the message that contains the necessary key to decrypt your message and read your information. If the symmetric key needed to decrypt your message is not protected, there is no use in encrypting the message in the first place. So we use an asymmetric algorithm to encrypt the symmetric key, as depicted in Figure 8-13. Why do we use the symmetric key on the message and the asymmetric key on the symmetric key? As stated earlier, the asymmetric algorithm takes longer because the math is more complex. Because your message is most likely going to be longer than the length of the key, we use the faster algorithm on the message (symmetric) and the slower algorithm on the key (asymmetric).

How does this actually work? Let's say Bill is sending Paul a message that Bill wants only Paul to be able to read. Bill encrypts his message with a secret key, so now Bill has ciphertext and a symmetric key. The key needs to be protected, so Bill encrypts the symmetric key with an asymmetric key. Remember that asymmetric algorithms use private and public keys, so Bill will encrypt the symmetric key with Paul's public key. Now Bill

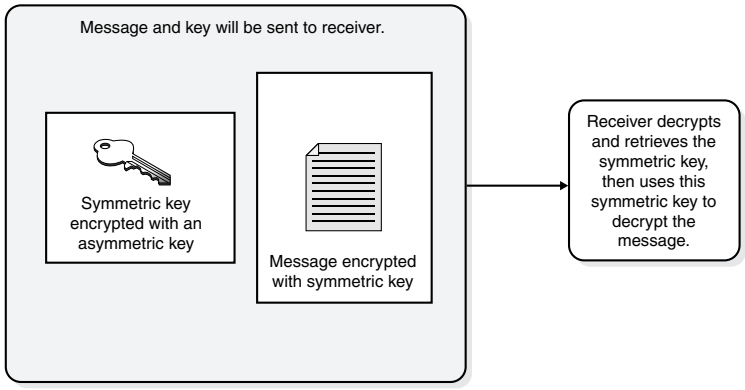
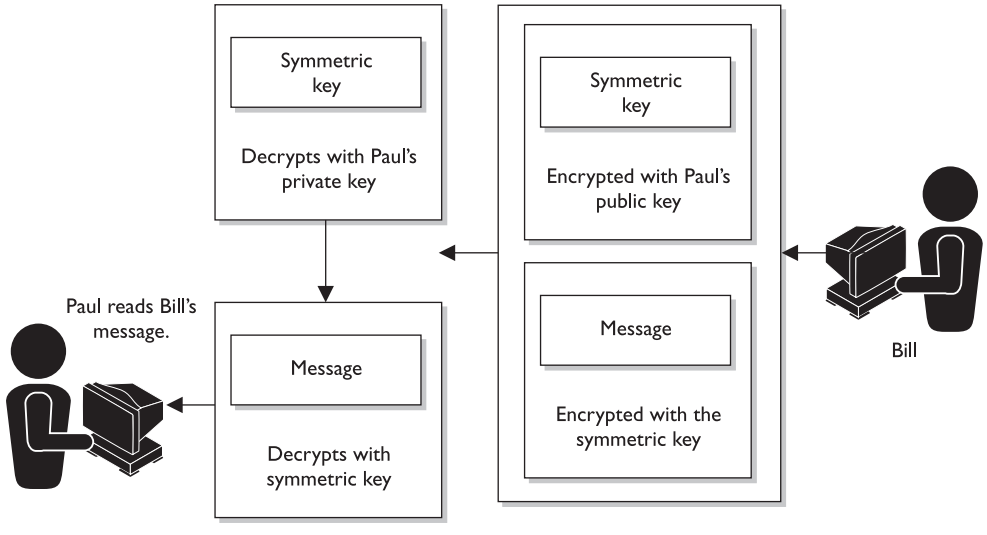


Figure 8-13 In a hybrid system, the asymmetric key is used to encrypt the symmetric key, and the symmetric key is used to encrypt the message.

has ciphertext from the message and ciphertext from the symmetric key. Why did Bill encrypt the symmetric key with Paul's public key instead of his own private key? Because if Bill encrypted it with his own private key, then anyone with Bill's public key could decrypt it and retrieve the symmetric key. However, Bill does not want anyone who has his public key to read his message to Paul. Bill only wants Paul to be able to read it. So Bill encrypts the symmetric key with Paul's public key. If Paul has done a good job protecting his private key, he will be the only one who can read Bill's message.



Paul receives Bill's message and Paul uses his private key to decrypt the symmetric key. Paul then uses the symmetric key to decrypt the message. Paul then reads Bill's very important and confidential message that asks Paul how his day is.

Now when I say that Bill is using this key to encrypt and that Paul is using that key to decrypt, those two individuals do not necessarily need to find the key on their hard drive and know how to properly apply it. We have software to do this for us—thank goodness.

If this is your first time with these issues and you are struggling, don't worry. I remember when I first started with these concepts, and they turned my brain into a pretzel. Just remember the following points:

- An asymmetric algorithm performs encryption and decryption by using public and private keys that are related to each other mathematically.
- A symmetric algorithm performs encryption and decryption by using a symmetric key.
- A symmetric key is used to encrypt the actual message.
- Public keys are used to encrypt the symmetric key for secure key exchange.
- A secret key is synonymous to a symmetric key.
- An asymmetric key refers to a public or private key.

So, that is how a hybrid system works. The symmetric algorithm creates a secret key that will be used to encrypt the bulk, or the message, and the asymmetric key encrypts the secret key.

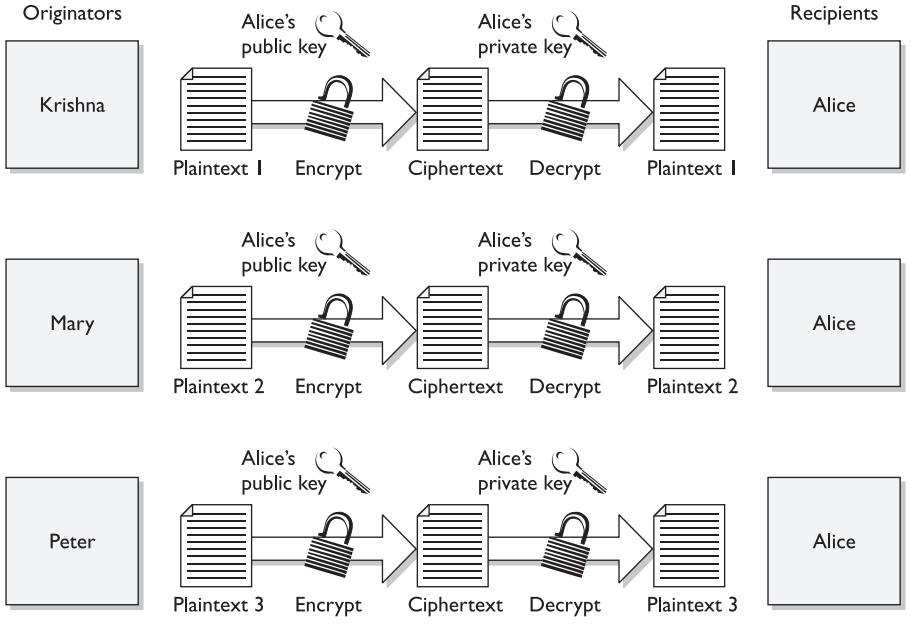
Now to ensure that some of these concepts are driven home, ask these questions of yourself without reading the answers provided:

1. If a symmetric key is encrypted with a receiver's public key, what security service(s) is provided?
2. If data are encrypted with the sender's private key, what security service(s) is provided?
3. If the sender encrypts data with the receiver's private key, what security services(s) is provided?
4. Why do we encrypt the message with the symmetric key?
5. Why don't we encrypt the symmetric key with another symmetric key?
6. What is the meaning of life?

Answers:

1. Confidentiality, because only the receiver's private key can be used to decrypt the symmetric key and only the receiver should have access to this private key.
2. Authenticity of the sender and nonrepudiation. If the receiver can decrypt the encrypted data with the sender's public key, then she knows the data was encrypted with the sender's private key.

3. None, because no one but the owner of the private key should have access to it. Trick question.
4. Because the asymmetric key is too slow.
5. We need to get the necessary symmetric key to the destination securely, which can only be carried out through asymmetric cryptography through the use of public and private keys.
6. 42.



Session Keys

Hey, I have a disposable key!
Response: Amazing. Now go away.

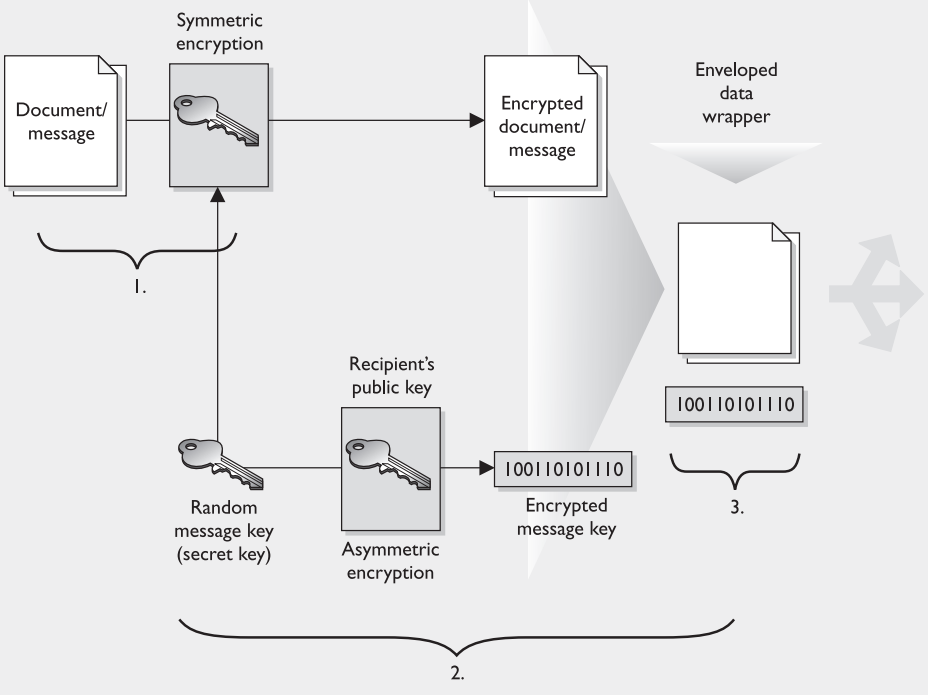
A ***session key*** is a symmetric key that is used to encrypt messages between two users. A session key is no different from the symmetric key described in the previous section, but it is only good for one communication session between users.

If Tanya has a symmetric key she uses to always encrypt messages between Lance and herself, then this symmetric key would not be regenerated or changed. They would use the same key every time they communicated using encryption. However, using the same key repeatedly increases the chances of the key being captured and the secure communication being compromised. If, on the other hand, a new symmetric key were

Digital Envelopes

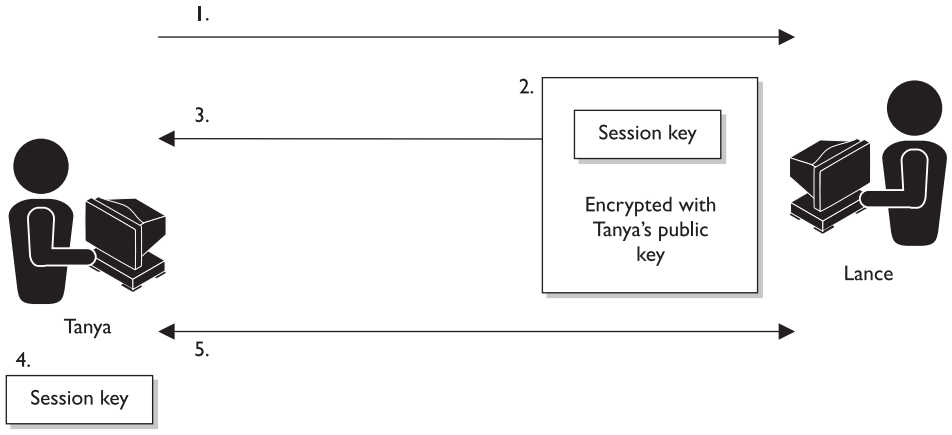
When cryptography is new to people, the process of using symmetric and asymmetric cryptography together can be a bit confusing. But it is important to understand these concepts, because they really are the core, fundamental concepts of all cryptography. This process is not just used in an e-mail client or in a couple of products—this is how it is done when data and a symmetric key must be protected in transmission.

The use of these two technologies together can be referred to as a hybrid approach, but more commonly as a ***digital envelope.***



generated each time Lance and Tanya wanted to communicate, as shown in Figure 8-14, it would be used only during their one dialog and then destroyed. If they wanted to communicate an hour later, a new session key would be created and shared.

A session key provides more protection than static symmetric keys because it is valid for only one session between two computers. If an attacker were able to capture the session key, she would have a very small window of time to use it to try to decrypt messages being passed back and forth.



1) Tanya sends Lance her public key.
2) Lance generates a random session key and encrypts it using Tanya's public key.
3) Lance sends the session key, encrypted with Tanya's public key, to Tanya.
4) Tanya decrypts Lance's message with her private key and now has a copy of the session key.
5) Tanya and Lance use this session key to encrypt and decrypt messages to each other.

Figure 8-14 A session key is generated so all messages can be encrypted during one particular session between users.

In cryptography, almost all data encryption takes place through the use of session keys. When you write an e-mail and encrypt it before sending it over the wire, it is actually being encrypted with a session key. If you write another message to the same person one minute later, a brand-new session key is created to encrypt that new message. So if an evildoer happens to figure out one session key, that does not mean she has access to all other messages you write and send off.

When two computers want to communicate using encryption, they must first go through a handshaking process. The two computers agree on the encryption algorithms that will be used and exchange the session key that will be used for data encryption. In a sense, the two computers set up a virtual connection between each other and are said to be in session. When this session is done, each computer tears down any data structures it built to enable this communication to take place, releases the resources, and destroys the session key. These things are taken care of by operating systems and applications in the background, so a user would not necessarily need to be worried about using the wrong type of key for the wrong reason. The software will handle this, but it is important for security professionals to understand the difference between the key types and the issues that surround them.

NOTE Private and symmetric keys should not be available in cleartext. This may seem obvious to you, but there have been several implementations over time that have allowed for this type of compromise to take place.

Wireless Security Woes

We covered the different 802.11 standards and the Wired Equivalent Privacy (WEP) protocol in Chapter 7. Among the long laundry list of security problems with WEP, not using unique session keys for data encryption is one of them. If only WEP is being used to encrypt wireless traffic, then in most implementations, just one static symmetric key is being used over and over again to encrypt the packets. This is one of the changes and advancements in the 802.11i standard, which makes sure each packet is encrypted with a unique session key.

Unfortunately, we don't always seem to be able to call an apple an apple. In many types of technology, the exact same thing can have more than one name. This could be because the different inventors of the technology had schizophrenia or it could mean that different terms just evolved over time that overlapped. Sadly, you could see symmetric cryptography referred to as one of the following labels:

- Single key cryptography
- Secret key cryptography
- Session key cryptography
- Private key cryptography

We know the difference between secret keys (static) and session keys (dynamic), but what is this "single key" and "private key" mess? Well, using the term "single key" makes sense, because the sender and receiver are using one single key. I (the author) am saddened that the term "private key" can be used to describe symmetric cryptography because it only adds more confusion to the difference between symmetric cryptography (where one symmetric key is used) and asymmetric cryptography (where both a private and public key are used). But no one asked or cares about my opinion, so we just need to remember this little quirk and still understand the difference between symmetric and asymmetric cryptography.

Types of Symmetric Systems

Several types of symmetric algorithms are used today. They have different methods of providing encryption and decryption functionality. The one thing they all have in common is that they are symmetric algorithms, meaning the sender and receiver are using two instances of the same key.

In this section, we will be walking through many of the following algorithms and their characteristics:

- Data Encryption Standard (DES)
- 3DES (Triple DES)
- Blowfish

- Twofish
- IDEA (International Data Encryption Algorithm)
- RC4, RC5, RC6
- AES
- SAFER
- Serpent

Data Encryption Standard

Data Encryption Standard (DES) has had a long and rich history within the computer community. The National Institute of Standards and Technology (NIST) researched the need for the protection of sensitive but unclassified data during the 1960s and initiated a cryptography program in the early 1970s. NIST invited vendors to submit data encryption algorithms to be used as a cryptographic standard. IBM had already been developing encryption algorithms to protect financial transactions. In 1974, IBM's 128-bit algorithm, named ***Lucifer,*** was submitted and accepted. The NSA modified this algorithm to use a key size of 64 bits (with eight bits used for parity, resulting in an effective key length of 56 bits) instead of the original 128 bits, and named it the ***Data Encryption Algorithm (DEA).*** Controversy arose about whether the NSA weakened Lucifer on purpose to enable it to decrypt messages not intended for it, but in the end the modified Lucifer became a national cryptographic standard in 1977 and an American National Standards Institute (ANSI) standard in 1978.

NOTE DEA is the algorithm that fulfills DES, which is really just a standard. So DES is the standard and DEA is the algorithm, but in the industry we usually just refer to it as DES. The CISSP exam may refer to the algorithm by either name, so remember both.

DES has been implemented in a majority of commercial products using cryptography functionality and in the applications of almost all government agencies. It was tested and approved as one of the strongest and most efficient cryptographic algorithms available. The continued overwhelming support of the algorithm is what caused the most confusion when the NSA announced in 1986 that, as of January 1988, the agency would no longer endorse DES and that DES-based products would no longer fall under compliance with Federal Standard 1027. The NSA felt that because DES had been so popular for so long, it would surely be targeted for penetration and become useless as an official standard. Many researchers disagreed, but NSA wanted to move on to a newer, more secure, and less popular algorithm as the new standard.

The NSA's decision to drop its support for DES caused major concern and negative feedback. At that time, it was shown that DES still provided the necessary level of protection; that projections estimated a computer would require thousands of years to crack DES; that DES was already embedded in thousands of products; and that there was no equivalent substitute. NSA reconsidered its decision and NIST ended up recertifying DES for another five years.

In 1998, the Electronic Frontier Foundation built a computer system for $250,000 that broke DES in three days using a brute force attack against the keyspace. It contained 1536 microprocessors running at 40MHz, which performed 60 million test decryptions per second per chip. Although most people do not have these types of systems to conduct such attacks, as Moore's Law holds true and microprocessors increase in processing power, this type of attack will become more feasible for the average attacker. This brought about 3DES, which provides stronger protection, as discussed later in the chapter.

DES was later replaced by the Rijndael algorithm as the ***Advanced Encryption Standard (AES)*** by NIST. This means that Rijndael is the new approved method of encrypting sensitive but unclassified information for the U.S. government; it has been accepted by, and is widely used in, the public arena today.

How Does DES Work?

How does DES work again?
Response: With voodoo magic and a dead chicken.

DES is a symmetric block encryption algorithm. When 64-bit blocks of plaintext go in, 64-bit blocks of ciphertext come out. It is also a symmetric algorithm, meaning the same key is used for encryption and decryption. It uses a 64-bit key: 56 bits make up the true key, and eight bits are used for parity.

When the DES algorithm is applied to data, it divides the message into blocks and operates on them one at a time. The blocks are put through 16 rounds of transposition and substitution functions. The order and type of transposition and substitution functions depend on the value of the key used with the algorithm. The result is 64-bit blocks of ciphertext.

What Does It Mean When an Algorithm Is Broken?

I dropped my algorithm.
Response: Well, now it's broken.

As described in an earlier section, DES was finally broken with a dedicated computer lovingly named the DES Cracker. But what does "broken" really mean?

In most instances, an algorithm is broken if someone is able to uncover a key that was used during an encryption process. So let's say Kevin encrypted a message and sent it to Valerie. Marc captures this encrypted message and carries out a brute force attack on it, which means he tries to decrypt the message with different keys until he uncovers the right one. Once he identifies this key, the algorithm is considered broken. So does that mean the algorithm is worthless? It depends upon who your enemies are.

If an algorithm is broken through a brute force attack, this just means the attacker identified the one key that was used for one instance of encryption. But in proper implementations, we should be encrypting data with session keys, which are good only for that one session. So even if the attacker uncovers one session key, it may be useless to the attacker, in which case he now has to work to identify a new session key.

If your information is of sufficient value that enemies or thieves would exert a lot of resources to break the encryption (as may be the case for financial transactions or military secrets), you would not use an algorithm that has been broken. If you are encrypting messages to your mother about a meatloaf recipe, you likely are not going to worry about whether the algorithm has been broken.

So breaking an algorithm can take place through brute force attacks or by identifying weaknesses in the algorithm itself. Brute force attacks have increased in potency because of the increased processing capacity of computers today. An algorithm that uses a 40-bit key has around 1 trillion possible key values. If a 56-bit key is used, then there are approximately 72 quadrillion different key values. This may seem like a lot, but relative to today's computing power, these key sizes do not provide much protection at all.

On a final note, algorithms are built on the current understanding of mathematics. As the human race advances in mathematics, the level of protection that today's algorithms provide may crumble.

DES Modes

Block ciphers have several modes of operation. Each mode specifies how a block cipher will operate. One mode may work better in one type of environment for specific functionality, whereas another mode may work better in another environment with totally different requirements. It is important that vendors who employ DES (or any block cipher) understand the different modes and which one to use for which purpose.

DES and other symmetric block ciphers have several distinct modes of operation that are used in different situations for different results. You just need to understand five of them:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter Mode (CTR)

Electronic Code Book Mode ECB mode operates like a code book. A 64-bit data block is entered into the algorithm with a key, and a block of ciphertext is produced. For a given block of plaintext and a given key, the same block of ciphertext is always produced. Not all messages end up in neat and tidy 64-bit blocks, so ECB incorporates padding to address this problem. ECB is the easiest and fastest mode to use, but as we will see, it has its dangers.

A key is basically instructions for the use of a code book that dictates how a block of text will be encrypted and decrypted. The code book provides the recipe of substitutions and permutations that will be performed on the block of plaintext. The security issue that comes up with using ECB mode is that each block will be encrypted with the exact same key, and thus the exact same code book. So, two bad things can happen here: an attacker could uncover the key and thus have the key to decrypt all the blocks of data, or an attacker could gather the ciphertext and plaintext of each block and build the code book that was used, without needing the key.

The crux of the problem is that there is not enough randomness to the process of encrypting the independent blocks, so if this mode is used to encrypt a large amount of data, it could be cracked more easily than the other modes that block ciphers can work in. So the next question to ask is, why even use this mode? This mode is the fastest and easiest, so we use it to encrypt small amounts of data, such as PINs, challenge-response values in authentication processes, and encrypting keys.

Because this mode works with blocks of data independently, data within a file do not have to be encrypted in a certain order. This is very helpful when using encryption in databases. A database has different pieces of data accessed in a random fashion. If it is encrypted in ECB mode, then any record or table can be added, encrypted, deleted, or decrypted independently of any other table or record. Other DES modes are dependent upon the text encrypted before them. This dependency makes it harder to encrypt and decrypt smaller amounts of text, because the previous encrypted text would need to be decrypted first. (Once we cover chaining in the next section, this dependency will make more sense.)

Because ECB mode does not use chaining, you should not use it to encrypt large amounts of data, because patterns would eventually show themselves.

Cipher Block Chaining Mode In ECB mode, a block of plaintext and a key will always give the same ciphertext. This means that if the word "balloon" were encrypted and the resulting ciphertext were "hwicssn," each time it was encrypted using the same key, the same ciphertext would always be given. This can show evidence of a pattern, enabling an evildoer, with some effort, to discover the pattern and get a step closer to compromising the encryption process.

Cipher Block Chaining (CBC) does not reveal a pattern, because each block of text, the key, and the value based on the previous block are processed in the algorithm and applied to the next block of text, as shown in Figure 8-15. This results in more random ciphertext. Ciphertext is extracted and used from the previous block of text. This provides dependence between the blocks, in a sense chaining them together. This is where the name Cipher Block Chaining comes from, and it is this chaining effect that hides any patterns.

The results of one block are XORed with the next block before it is encrypted, meaning each block is used to modify the following block. This chaining effect means that a particular ciphertext block is dependent upon all blocks before it, not just the previous block.

As an analogy, let's say you have five buckets of marbles. Each bucket contains a specific color of marbles: red, blue, yellow, black, and green. The first bucket of red marbles (block of bits) you shake and tumble around (encrypt) to get them all mixed up. Then

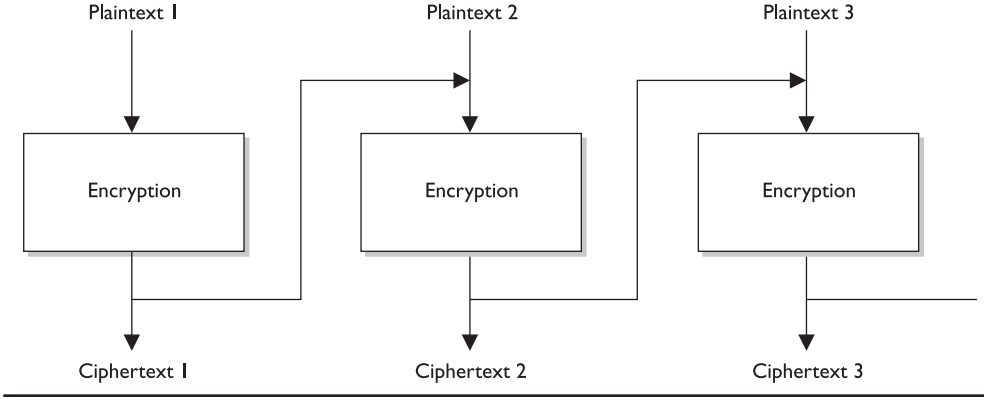


Figure 8-15 In CBC mode, the ciphertext from the previous block of data is used in encrypting the next block of data.

you take the second bucket of marbles, which are blue, and pour in the red marbles and go through the same exercise of shaking and tumbling them. You pour this bucket of marbles into your next bucket and shake them all up. This illustrates the incorporated randomness that is added when using chaining in a block encryption process.

When we encrypt our very first block using CBC, we do not have a previous block of ciphertext to "dump in" and use to add the necessary randomness to the encryption process. If we do not add a piece of randomness when encrypting this first block, then the bad guys could identify patterns, work backward, and uncover the key. So, we use an initialization vector (IVs were introduced previously in the "Initialization Vectors" section). The 64-bit IV is XORed with the first block of plaintext and then it goes through its encryption process. The result of that (ciphertext) is XORed with the second block of plaintext, and then the second block is encrypted. This continues for the whole message. It is the chaining that adds the necessary randomness that allows us to use CBC mode to encrypt large files. Neither the individual blocks nor the whole message will show patterns that will allow an attacker to reverse-engineer and uncover the key.

If we choose a different IV each time we encrypt a message, even if it is the same message, the ciphertext will always be unique. This means that if you send the same message out to 50 people and encrypt each one using a different IV, the ciphertext for each message will be different. Pretty nifty.

Cipher Feedback Mode Sometimes block ciphers can emulate a stream cipher. Before we dig into how this would happen, let's first look at why. If you are going to send an encrypted e-mail to your boss, your e-mail client will use a symmetric block cipher working in CBC mode. The e-mail client would not use ECB mode, because most messages are long enough to show patterns that can be used to reverse-engineer the process and uncover the encryption key. The CBC mode is great to use when you need to send large chunks of data at a time. But what if you are not sending large chunks of data at one time, but instead are sending a steady stream of data to a destination? If you are working on a terminal that communicates with a back-end terminal server, what is really going on is that each keystroke and mouse movement you make is sent to the backend server in chunks of eight bits to be processed. So even though it seems as though the computer you are working on is carrying out your commands and doing the processing you are requesting, it is not—this is happening on the server. Thus, if you need to encrypt the data that go from your terminal to the terminal server, you could not use CBC mode because it only encrypts blocks of data 64 bits in size. You have blocks of eight bits you need to encrypt. So what do you know? We have just the mode for this type of situation!

Figure 8-16 illustrates how CFB mode works, which is really a combination of a block cipher and a stream cipher. For the first block of eight bits that needs to be encrypted, we do the same thing we did in CBC mode, which is to use an IV. Recall how stream ciphers work: The key and the IV are used by the algorithm to create a keystream, which is just a random set of bits. This set of bits is XORed to the block of plaintext, which results in the same size block of ciphertext. So the first block (eight bits) is XORed to the set of bits created through the keystream generator. Two things take place with this resulting eight-bit block of ciphertext. One copy goes over the wire to the destination (in our scenario, to the terminal server) and another copy is used to encrypt the next block of eight-bit plaintext. Adding this copy of ciphertext to the encryption process of the next block adds more randomness to the encryption process.

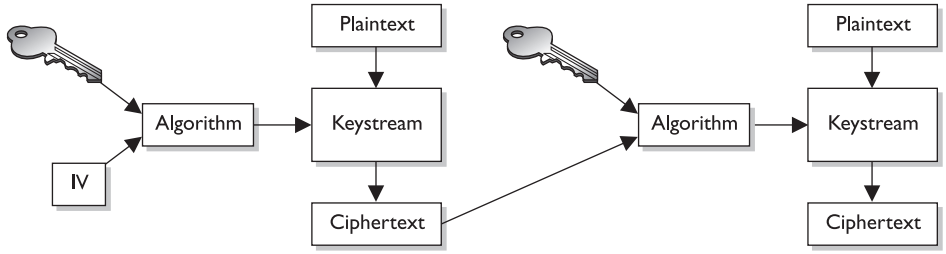


Figure 8-16 A block cipher working in CFB mode

We walked through a scenario where eight-bit blocks needed to be encrypted, but in reality CFB mode can be used to encrypt any size blocks, even blocks of just one bit. But since most of our encoding maps eight bits to one character, using CFB to encrypt eight-bit blocks is very common.

NOTE When using CBC mode, it is a good idea to use a unique IV value per message, but this is not necessary since the message being encrypted is usually very large. When using CFB mode, we are encrypting a smaller amount of data, so it is imperative a new IV value be used to encrypt each new stream of data.

Output Feedback Mode As you have read, you can use ECB mode for the process of encrypting small amounts of data, such as a key or PIN value. These components will be around 64 bits or more, so ECB mode works as a true block cipher. You can use CBC mode to encrypt larger amounts of data, in block sizes of 64 bits. In situations where you need to encrypt a smaller amount of data, you need the cipher to work like a stream cipher and encrypt individual bits of the blocks, as in CFB. In some cases, you still need to encrypt a small amount of data at a time (one to eight bits), but you need to ensure possible errors do not affect your encryption and decryption processes.

If you look back at Figure 8-16, you see that the ciphertext from the previous block is used to encrypt the next block of plaintext. What if a bit in the first ciphertext gets corrupted? Then we have corrupted values going into the process of encrypting the next block of plaintext, and this problem just continues because of the use of chaining in this mode. Now look at Figure 8-17. It looks terribly similar to Figure 8-16, but notice that the values used to encrypt the next block of plaintext are coming directly from the keystream, not from the resulting ciphertext. This is the difference between the two modes.

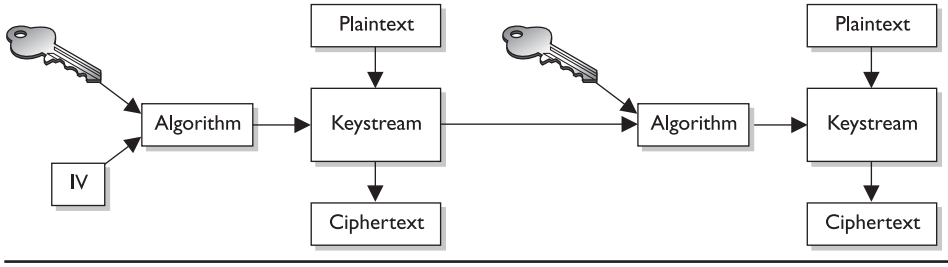


Figure 8-17 A block cipher working in OFB mode

If you need to encrypt something that would be very sensitive to these types of errors, such as digitized video or digitized voice signals, you should not use CFB mode. You should use OFB mode instead, which reduces the chance that these types of bit corruptions can take place.

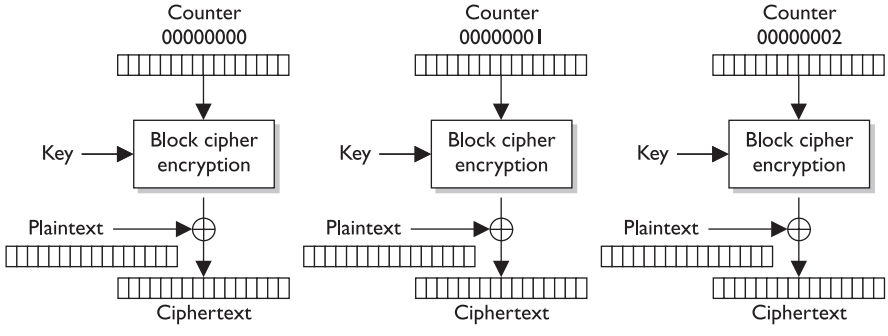
So OFB is a mode that a block cipher can work in when it needs to emulate a stream, because it encrypts small amounts of data at a time, but it has a smaller chance of creating and extending errors throughout the full encryption process.

To ensure OFB and CFB are providing the most protection possible, the size of the ciphertext (in CFB) or keystream values (in OFB) needs to be the same size as the block of plaintext being encrypted. This means that if you are using CFB and are encrypting eight bits at a time, the ciphertext you bring forward from the previous encryption block needs to be eight bits. Otherwise, you are repeating values over and over, which introduces patterns. (This is the same reason why a one-time pad should be used only one time and should be as long as the message itself.)

Counter Mode Counter Mode (CTR) is very similar to OFB mode, but instead of using a randomly unique IV value to generate the keystream values, this mode uses an IV counter that increments for each plaintext block that needs to be encrypted. The unique counter ensures that each block is XORed with a unique keystream value.

The other difference is that there is no chaining involved, which means no ciphertext is brought forward to encrypt the next block. Since there is no chaining, the encryption of the individual blocks can happen in parallel, which increases the performance. The main reason CTR would be used instead of the other modes is performance.

This mode has been around for quite some time and is used in encrypting ATM cells for virtual circuits, in IPSec, and is now integrated in the new wireless security standard, 802.11i. A developer would choose to use this mode in these situations because individual ATM cells or packets going through an IPSec tunnel or over radio frequencies may not arrive at the destination in order. Since chaining is not involved, the destination can decrypt and begin processing the packets without having to wait for the full message to arrive and *then* decrypt all the data.



References

- **"Cryptography," by Bill Unruh** http://axion.physics.ubc.ca/crypt.html
- **Cryptography Research, Inc. home page** www.cryptography.com

Here We Are
If this is your first time trying to understand cryptography, you may be exasperated by now. Don't get too uptight. Many people are new to cryptography, because all of this magic just seems to work in the background without us having to understand it or mess with it. If you need more foundational understanding, check out the preceding references

Triple-DES

We went from DES to Triple-DES (3DES), so it might seem we skipped Double-DES. We did. Double-DES has a key length of 112 bits, but there is a specific attack against Double-DES that reduces its work factor to about the same as DES. Thus, it is no more secure than DES. So let's move on to 3DES.

Many successful attacks against DES and the realization that the useful lifetime of DES was about up brought much support for 3DES. NIST knew that a new standard had to be created, which ended up being AES, but a quick fix was needed in the meantime to provide more protection for sensitive data. The result: 3DES.

3DES uses 48 rounds in its computation, which makes it highly resistant to differential cryptanalysis. However, because of the extra work 3DES performs, there is a heavy performance hit. It can take up to three times longer than DES to perform encryption and decryption.

Although NIST has selected the Rijndael algorithm to replace DES as the AES, NIST and others expect 3DES to be around and used for quite some time.

3DES can work in different modes, and the mode chosen dictates the number of keys used and what functions are carried out:

- **DES-EEE3** Uses three different keys for encryption, and the data are encrypted, encrypted, encrypted.
- **DES-EDE3** Uses three different keys for encryption, and the data are encrypted, decrypted, and encrypted.
- **DES-EEE2** The same as DES-EEE3 but uses only two keys, and the first and third encryption processes use the same key.
- **DES-EDE2** The same as DES-EDE3 but uses only two keys, and the first and third encryption processes use the same key.

EDE may seem a little odd at first. How much protection could be provided by encrypting something, decrypting it, and encrypting it again? The decrypting portion here is decrypted with a different key. When data are encrypted with one symmetric key and decrypted with a different symmetric key, it is jumbled even more. So the data are not actually decrypted in the middle function, they are just run through a decryption process with a different key. Pretty tricky.

The Advanced Encryption Standard

After DES was used as an encryption standard for over 20 years and it was cracked in a relatively short time once the necessary technology was available, NIST decided a new standard, the Advanced Encryption Standard (AES), needed to be put into place. In

January 1997, NIST announced its request for AES candidates and outlined the requirements in FIPS PUB 197. AES was to be a symmetric block cipher supporting key sizes of 128, 192, and 256 bits. The following five algorithms were the finalists:

- **MARS** Developed by the IBM team that created Lucifer
- **RC6** Developed by RSA Laboratories
- **Serpent** Developed by Ross Anderson, Eli Biham, and Lars Knudsen
- **Twofish** Developed by Counterpane Systems
- **Rijndael** Developed by Joan Daemen and Vincent Rijmen

Out of these contestants, Rijndael was chosen. The block sizes that Rijndael supports are 128, 192, and 256 bits. The number of rounds depends upon the size of the block and the key length:

- If both the key and block size are 128 bits, there are 10 rounds.
- If both the key and block size are 192 bits, there are 12 rounds.
- If both the key and block size are 256 bits, there are 14 rounds.

Rijndael works well when implemented in software and hardware in a wide range of products and environments. It has low memory requirements and has been constructed to easily defend against timing attacks.

Rijndael was NIST's choice to replace DES. It is now the algorithm required to protect sensitive but unclassified U.S. government information.

NOTE DEA is the algorithm used within DES, and Rijndael is the algorithm used in AES. In the industry, we refer to these as DES and AES instead of the actual algorithms.

International Data Encryption Algorithm

International Data Encryption Algorithm (IDEA) is a block cipher and operates on 64-bit blocks of data. The 64-bit data block is divided into 16 smaller blocks, and each has eight rounds of mathematical functions performed on it. The key is 128 bits long and IDEA is faster than DES when implemented in software.

The IDEA algorithm offers different modes similar to the modes described in the DES section, but it is considered to be harder to break than DES because it has a longer key size. IDEA is used in the PGP and other encryption software implementations. It was thought to replace DES, but it is patented, meaning that licensing fees would have to be paid to use it.

As of this writing, there have been no successful practical attacks against this algorithm, although there have been numerous attempts.

Blowfish

One fish, two fish, red fish, blowfish.
Response: Hmm, I thought it was blue fish.

Blowfish is a block cipher that works on 64-bit blocks of data. The key length can be anywhere from 32 bits up to 448 bits, and the data blocks go through 16 rounds of

cryptographic functions. Bruce Schneier designed it. It was intended as a replacement to the aging DES. While many of the other algorithms have been proprietary in nature and thus encumbered by patents or kept as government secrets, this wasn't the case with Blowfish. Bruce Schneier, the creator of Blowfish, has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone."

RC4

RC4 is one of the most commonly implemented stream ciphers. It has a variable key size, is used in the SSL protocol, and was (improperly) implemented in the 802.11 WEP protocol standard. RC4 was developed in 1987 by Ron Rivest and was considered a trade secret of RSA Data Security, Inc. until someone posted the source code on a mailing list. Since the source code was released nefariously, the stolen algorithm is sometimes implemented and referred to as ArcFour or ARC4 because the title RC4 is trademarked.

The algorithm is very simple, fast, and efficient, which is why it became so popular.

RC5

RC5 is a block cipher that has a variety of parameters it can use for block size, key size, and the number of rounds used. It was created by Ron Rivest and analyzed by RSA Data Security, Inc. The block sizes used in this algorithm are 32, 64, or 128 bits, and the key size goes up to 2048 bits. The number of rounds used for encryption and decryption is also variable. The number of rounds can go up to 255.

RC6

RC6 is a block cipher that was built upon RC5, so it has all the same attributes as RC5. The algorithm was developed mainly to be submitted as AES, but Rijndael was chosen instead. There were some modifications of the RC5 algorithm to increase the overall speed, the result of which is RC6.

Cryptography Notation

In some resources, you may run across rc5-w/r/b or RC5-32/12/16. This is a type of shorthand that describes the configuration of the algorithm.

- w = Word size, in bits, which can be 16, 32, or 64 bits in length
- r = Number of rounds, which can be 0 to 255

So RC5-32/12/16 would mean the following:

- 32-bit words, which means it encrypts 64-bit data blocks
- Using 12 rounds
- With a 16-byte (128-bit) key

A developer configures these parameters (words, number of rounds, key size) for the algorithm for specific implementations. The existence of these parameters gives developers extensive flexibility.

Types of Asymmetric Systems

As described earlier in the chapter, using purely symmetric key cryptography has three drawbacks, which affect the following:

- **Security services** Purely symmetric key cryptography provides confidentiality only, not authentication or nonrepudiation.
- **Scalability** As the number of people who need to communicate increases, so does the number of symmetric keys required, meaning more keys must be managed.
- **Secure key distribution** The symmetric key must be delivered to its destination through a secure courier.

Despite these drawbacks, symmetric key cryptography was all that the computing society had available for encryption for quite some time. Symmetric and asymmetric cryptography did not arrive in this world on the same day or even in the same decade. We dealt with the issues surrounding symmetric cryptography for quite some time, waiting for someone smarter to come along and save us from some of this grief.

The Diffie-Hellman Algorithm

The first group to address the shortfalls of symmetric key cryptography decided to attack the issue of secure distribution of the symmetric key. Whitfield Diffie and Martin Hellman worked on this problem and ended up developing the first asymmetric key agreement algorithm, called, naturally, Diffie-Hellman.

To understand how Diffie-Hellman works, consider an example. Let's say that Tanya and Erika would like to communicate over an encrypted channel by using Diffie-Hellman. They would both generate a private and public key pair and exchange public keys. Tanya's software would take her private key (which is just a numeric value) and Erika's public key (another numeric value) and put them through the Diffie-Hellman algorithm. Erika's software would take her private key and Tanya's public key and insert them into the Diffie-Hellman algorithm on her computer. Through this process, Tanya and Erika derive the same shared value, which is used to create instances of symmetric keys.

So, Tanya and Erika exchanged information that did not need to be protected (their public keys) over an untrusted network, and in turn generated the exact same symmetric key on each system. They both can now use these symmetric keys to encrypt, transmit, and decrypt information as they communicate with each other.

NOTE The preceding example describes key agreement, which is different from key exchange, the functionality used by the other asymmetric algorithms that will be discussed in this chapter. With key exchange functionality, the sender encrypts the symmetric key with the receiver's public key before transmission.

The Diffie-Hellman algorithm enables two systems to receive a symmetric key securely without requiring a previous relationship or prior arrangements. The algorithm allows for key distribution, but does not provide encryption or digital signature functionality. The algorithm is based on the difficulty of calculating discrete logarithms in a finite field.

Diffie-Hellman Mathematical Steps

1. Tanya chooses a large random integer (x) and sends it to Erika.
2. Erika also chooses a large random integer (y) and sends it to Tanya.
3. Tanya's software computes the following:
   $K = Y^x \bmod n$
4. Erika's software computes the following:
   $K = X^y \bmod n$

Erika's and Tanya's results are the same (K), which is the symmetric key they can now use to encrypt data they will send back and forth.

The Diffie-Hellman algorithm is vulnerable to a man-in-the-middle attack, because no authentication occurs before public keys are exchanged. In our example, when Tanya sends her public key to Erika, how does Erika really know it is Tanya's public key? What if Lance spoofed his identity, told Erika he was Tanya, and sent over his key? Erika would accept this key, thinking it came from Tanya. Let's walk through the steps of how this type of attack would take place, as illustrated in Figure 8-18:

1. Tanya sends her public key to Erika, but Lance grabs the key during transmission so it never makes it to Erika.
2. Lance spoofs Tanya's identity and sends over his public key to Erika. Erika now thinks she has Tanya's public key.
3. Erika sends her public key to Tanya, but Lance grabs the key during transmission so it never makes it to Tanya.

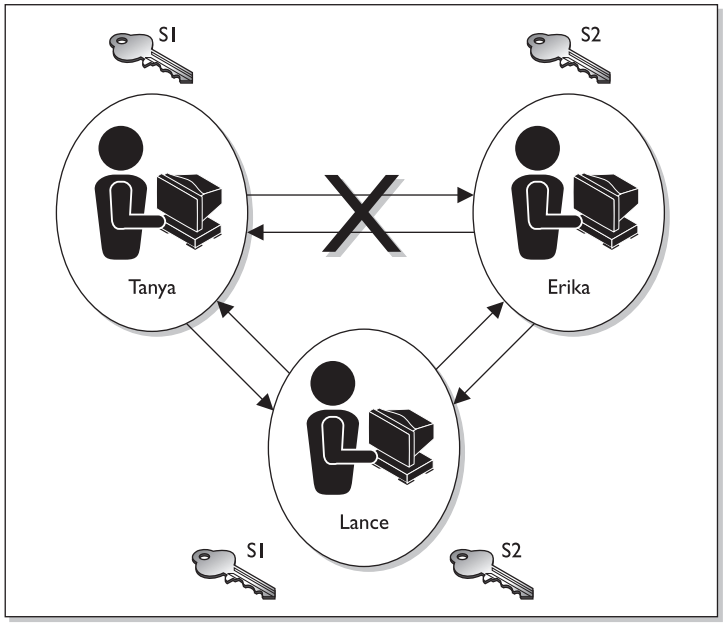


Figure 8-18
A man-in-the-middle attack

4. Lance spoofs Erika's identity and sends over his public key to Tanya. Tanya now thinks she has Erika's public key.
5. Tanya combines her private key and Lance's public key and creates symmetric key S1.
6. Lance combines his private key and Tanya's public key and creates symmetric key S1.
7. Erika combines her private key and Lance's public key and creates symmetric key S2.
8. Lance combines his private key and Erika's public key and creates symmetric key S2.
9. Now Tanya and Lance share a symmetric key (S1) and Erika and Lance share a different symmetric key (S2). Tanya and Erika think they are sharing a key between themselves and do not realize Lance is involved.
10. Tanya writes a message to Erika, uses her symmetric key (S1) to encrypt the message, and sends it.
11. Lance grabs the message and decrypts it with symmetric key S1, reads or modifies the message and re-encrypts it with symmetric key S2, and then sends it to Erika.
12. Erika takes symmetric key S2 and uses it to decrypt and read the message.

The countermeasure to this type of attack is to have authentication take place before accepting someone's public key, which usually happens through the use of digital signatures and digital certificates.

NOTE Although the Diffie-Hellman algorithm is vulnerable to a man-in-the-middle attack, it does not mean this type of compromise can take place anywhere this algorithm is deployed. Most implementations include another piece of software or a protocol that compensates for this vulnerability. But some do not. As a security professional, you should understand these issues.

RSA

Give me an R, *give me an* S, *give me an… How do you spell "RSA" again?*
Response: Sigh.

RSA, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, is a public key algorithm that is the most popular when it comes to asymmetric algorithms. RSA is a worldwide de facto standard and can be used for digital signatures, key exchange, and encryption. It was developed in 1978 at MIT and provides authentication as well as key encryption.

The security of this algorithm comes from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large prime numbers, and the necessary activity required to decrypt a message from ciphertext to plaintext using a private key is comparable to factoring a product into two prime numbers.

What Is the Difference Between Public Key Cryptography and Public Key Infrastructure?

Public key cryptography is the use of an asymmetric algorithm. Thus, the terms asymmetric algorithm and public key cryptography are interchangeable and mean the same thing. Examples of asymmetric algorithms are RSA, elliptic curve cryptosystem (ECC), Diffie-Hellman, El Gamal, LUC, and Knapsack. These algorithms are used to create public/private key pairs, perform key exchange or agreement, and generate and verify digital signatures.

Note that Diffie-Hellman can only perform key agreement and cannot generate or verify digital signatures.

Public key infrastructure (PKI) is a different animal. It is not an algorithm, a protocol, or an application—it is an infrastructure based on public key cryptography. Let's look at why we even need PKIs today. When Erika needs to send Tanya a symmetric key securely, she must obtain Tanya's public key. Erika could get the key from a public repository that holds public keys for many individuals, but if Lance has switched out Tanya's public key and inserted his own, when Erika acquires a key she thinks is Tanya's, she actually receives Lance's key and Erika has no idea.

NOTE A prime number is a positive whole number with no proper divisors, meaning the only numbers that can divide a prime number are 1 and the number itself.

One advantage of using RSA is that it can be used for encryption and digital signatures. Using its one-way function, RSA provides encryption and signature verification, and the inverse direction performs decryption and signature generation.

RSA has been implemented in applications; operating systems by Microsoft, Apple, Sun, and Novell; and at the hardware level in network interface cards, secure telephones, and smart cards. It can be used as a *key exchange protocol,* meaning it is used to encrypt the symmetric key to get it securely to its destination. RSA has been most commonly used with the symmetric algorithm DES, which is quickly being replaced with AES. So, when RSA is used as a key exchange protocol, a cryptosystem generates a symmetric key using either the DES or AES algorithm. Then the system encrypts the symmetric key with the receiver's public key and sends it to the receiver. The symmetric key is protected because only the individual with the corresponding private key can decrypt and extract the symmetric key.

Diving into Numbers

Cryptography is really all about using mathematics to scramble bits into an undecipherable form and then using the same mathematics in reverse to put the bits back into a form that can be understood by computers and people. RSA's mathematics are based on the difficulty of factoring a large integer into its two prime factors. Put on your nerdy hat with the propeller and let's look at how this algorithm works.

The algorithm creates a public key and a private key from a function of large prime numbers. When data are encrypted with a public key, only the corresponding private key can decrypt the data. This act of decryption is basically the same as factoring the product of two prime numbers. So, let's say I have a secret (encrypted message), and for you to be able to uncover the secret, you have to take a specific large number and factor it and come up with the two numbers I have written down on a piece of paper. This may sound simplistic, but the number you must properly factor can be 2^{300} in size. Not as easy as one may think.

The following sequence describes how the RSA algorithm comes up with the keys in the first place:

1. Choose two random large prime numbers, p and q.
2. Generate the product of these numbers: $n = pq$.
3. Choose a random number to be the encryption key, e. Make sure that e and $(p - 1)(q - 1)$ are relatively prime.
4. Compute the decryption key, d. This is $ed = 1 \bmod (p - 1)(q - 1)$ or $d = e^{-1} \bmod ([p - 1][q - 1])$.
5. The public key = (n, e).
6. The private key = d.
7. The original prime numbers p and q are discarded securely.

We now have our public and private keys, but how do they work together?

If you need to encrypt message m with your public key (e, n), the following formula is carried out:

$C = m^e \bmod n$

Then you need to decrypt the message with your private key (d), so the following formula is carried out:

$M = c^d \bmod n$

You may be thinking, "Well, I don't understand these formulas, but they look simple enough. Why wouldn't someone be able to break these small formulas and be able to uncover the encryption key?" Maybe someone will one day. As the human race advances in its understanding of mathematics and as processing power increases and cryptanalysis evolves, the RSA algorithm may be broken one day. If we figure out how to quickly and more easily factor large numbers into their original prime values, all of these cards fall down and this algorithm would no longer provide the security it does today. But we have not hit that bump in the road yet, so we are all happily using RSA in our computing activities.

One-Way Functions

A ***one-way function*** is a mathematical function that is easier to compute in one direction than in the opposite direction. An analogy of this is when you drop a glass on the floor. Although dropping a glass on the floor is easy, putting all the pieces back together again to reconstruct the original glass is next to impossible. This concept is simi-

lar to how a one-way function is used in cryptography, which is what the RSA algorithm, and all other asymmetric algorithms, is based upon.

The easy direction of computation in the one-way function that is used in the RSA algorithm is the process of multiplying two large prime numbers. Multiplying the two numbers to get the resulting product is much easier than factoring the product and recovering the two initial large prime numbers used to calculate the obtained product, which is the difficult direction. RSA is based on the difficulty of factoring large numbers that are the product of two large prime numbers. Attacks on these types of cryptosystems do not necessarily try every possible key value, but rather try to factor the large number, which will give the attacker the private key.

When a user encrypts a message with a public key, this message is encoded with a one-way function (breaking a glass). This function supplies a ***trapdoor*** (knowledge of how to put the glass back together), but the only way the trapdoor can be taken advantage of is if it is known about and the correct code is applied. The private key provides this service. The private key knows about the trapdoor, knows how to derive the original prime numbers, and has the necessary programming code to take advantage of this secret trapdoor to unlock the encoded message (reassembling the broken glass). Knowing about the trapdoor and having the correct functionality to take advantage of it are what make the private key private.

When a one-way function is carried out in the easy direction, encryption and digital signature verification functionality are available. When the one-way function is carried out in the hard direction, decryption and signature generation functionality are available. This means only the public key can carry out encryption and signature verification and only the private key can carry out decryption and signature generation.

As explained earlier in this chapter, *work factor* is the amount of time and resources it would take for someone to break an encryption method. In asymmetric algorithms, the work factor relates to the difference in time and effort that carrying out a one-way function in the easy direction takes compared to carrying out a one-way function in the hard direction. In most cases, the larger the key size, the longer it would take for the bad guy to carry out the one-way function in the hard direction (decrypt a message).

The crux of this section is that all asymmetric algorithms provide security by using mathematical equations that are easy to perform in one direction and next to impossible to perform in the other direction. The "hard" direction is based on a "hard" mathematical problem. RSA's hard mathematical problem requires factoring large numbers into their original prime numbers. Diffie-Hellman and El Gamal are based on the difficulty of calculating logarithms in a finite field.

El Gamal

El Gamal is a public key algorithm that can be used for digital signatures, encryption, and key exchange. It is based not on the difficulty of factoring large numbers but on calculating discrete logarithms in a finite field. El Gamal is actually an extension of the Diffie-Hellman algorithm.

Although El Gamal provides the same type of functionality as some of the other asymmetric algorithms, its main drawback is performance. When compared to other algorithms, this algorithm is usually the slowest.

Elliptic Curve Cryptosystems

Elliptic curves. That just sounds like fun.

Elliptic curves are rich mathematical structures that have shown usefulness in many different types of applications. An ***elliptic curve cryptosystem (ECC)*** provides much of the same functionality RSA provides: digital signatures, secure key distribution, and encryption. One differing factor is ECC's efficiency. ECC is more efficient than RSA and any other asymmetric algorithm.

Figure 8-19 is an example of an elliptic curve. In this field of mathematics, points on the curves comprise a structure called a group. These points are the values used in mathematical formulas for ECC's encryption and decryption processes. The algorithm computes discrete logarithms of elliptic curves, which is different from calculating discrete logarithms in a finite field (which is what Diffie-Hellman and El Gamal use).

Some devices have limited processing capacity, storage, power supply, and bandwidth, such as wireless devices and cellular telephones. With these types of devices, efficiency of resource use is very important. ECC provides encryption functionality, requiring a smaller percentage of the resources needed by RSA and other algorithms, so it is used in these types of devices.

In most cases, the longer the key, the more protection that is provided, but ECC can provide the same level of protection with a key size that is shorter than what RSA requires. Because longer keys require more resources to perform mathematical tasks, the smaller keys used in ECC require fewer resources of the device.

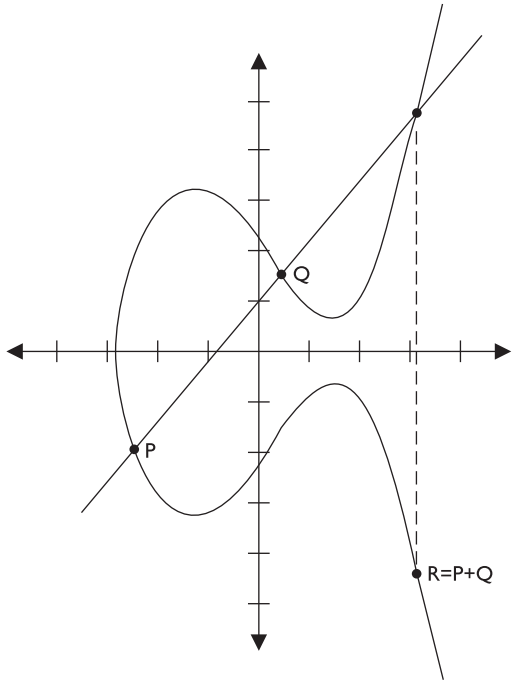


Figure 8-19
Elliptic curves

LUC

This algorithm is based on "Lucas sequences" and implements a discrete logarithm in a finite field, but by using the Lucas sequences, computation may take place more quickly.

Knapsack

Over the years, different versions of knapsack algorithms have arisen. The first to be developed, Merkle-Hellman, could be used only for encryption, but it was later improved upon to provide digital signature capabilities. These types of algorithms are based on the "knapsack problem," a mathematical dilemma that poses the following question: If you have several different items, each having its own weight, is it possible to add these items to a knapsack so the knapsack has a specific weight?

This algorithm was discovered to be insecure and is not currently used in cryptosystems.

Zero Knowledge Proof

Total knowledge zero. Yep, that's how I feel after reading all of this cryptography stuff!
Response: Just put your head between your knees and breathe slowly.

When military representatives are briefing the news media about some big world event, they have one goal in mind: to tell the story that the public is supposed to hear and nothing more. Do not provide extra information that someone could use to infer more information than they are supposed to know. The military has this goal because it knows that not just the good guys are watching CNN. This is an example of zero knowledge proof. You tell someone just the information they need to know without "giving up the farm."

Zero knowledge proof is used in cryptography also. If I encrypt something with my private key, you can verify my private key was used by decrypting the data with my public key. By encrypting something with my private key, I am proving to you I have my private key—but I do not give or show you my private key. I do not "give up the farm" by disclosing my private key. In a zero knowledge proof, the verifier cannot prove to another entity that this proof is real, because he does not have the private key to prove it. So, only the owner of the private key can prove he has possession of the key.

References

- **Cryptography information** www.cs.berkeley.edu/~daw/crypto.html
- **"Elliptic Curve Cryptography FAQ v1.12," by George Barwood (Dec. 12, 1997)** www.cryptoman.com/elliptic.htm

Message Integrity

My message was altered, thus it has character flaws.
Response: So do you.

Parity bits and cyclic redundancy check (CRC) functions have been used in protocols to detect modifications in streams of bits as they are passed from one computer to another, but they can usually only detect unintentional modifications. Unintentional

modifications can happen if a spike occurs in the power supply, if there is interference or attenuation on a wire, or if some other type of physical condition happens that causes the corruption of bits as they travel from one destination to another. Parity bits cannot identify whether a message was captured by an intruder, altered, and then sent on to the intended destination. The intruder can just recalculate a new parity value that includes his changes, and the receiver would never know the difference. For this type of protection, hash algorithms are required to successfully detect intentional and unintentional unauthorized modifications to data. We will now dive into hash algorithms and their characteristics.

The One-Way Hash

Now, how many times does the one-way hash run again?
Response: One, brainiac.

A ***one-way hash*** is a function that takes a variable-length string and a message and produces a fixed-length value called a hash value. For example, if Kevin wants to send a message to Maureen and he wants to ensure the message does not get altered in an unauthorized fashion while it is being transmitted, he would calculate a hash value for the message and append it to the message itself. When Maureen receives the message, she performs the same hashing function Kevin used and then compare her result with the hash value sent with the message. If the two values are the same, Maureen can be sure the message was not altered during transmission. If the two values are different, Maureen knows the message was altered, either intentionally or unintentionally, and she discards the message.

The hashing algorithm is not a secret—it is publicly known. The secrecy of the one-way hashing function is its "one-wayness." The function is run in only one direction, not the other direction. This is different from the one-way function used in public key cryptography, in which security is provided based on the fact that, without knowing a trapdoor, it is very hard to perform the one-way function backward on a message and come up with readable plaintext. However, one-way hash functions are never used in reverse; they create a hash value and call it a day. The receiver does not attempt to reverse the process at the other end, but instead runs the same hashing function one way and compares the two results.

The hashing one-way function takes place without the use of any keys. This means, for example, that if Cheryl writes a message, calculates a message digest, appends the digest to the message, and sends it on to Scott, Bruce can intercept this message, alter Cheryl's message, recalculate another message digest, append it to the message, and send it on to Scott. When Scott receives it, he verifies the message digest, but never knows the message was actually altered by Bruce. Scott thinks the message came straight from Cheryl and it was never modified, because the two message digest values are the same. If Cheryl wanted more protection than this, she would need to use ***message authentication code (MAC).***

A MAC function is an authentication scheme derived by applying a secret key to a message in some form. This does not mean the symmetric key is used to encrypt the

message, though. You should be aware of two basic types of MACs: a hash MAC (HMAC) and a CBC-MAC.

HMAC

In the previous example, if Cheryl were to use an HMAC function instead of just a plain hashing algorithm, a symmetric key would be concatenated with her message. The result of this process would be put through a hashing algorithm, and the result would be a MAC value. This MAC value is then appended to her message and sent to Scott. If Bruce were to intercept this message and modify it, he would not have the necessary symmetric key to create the MAC value that Scott will attempt to generate. Figure 8-20 walks through these steps.

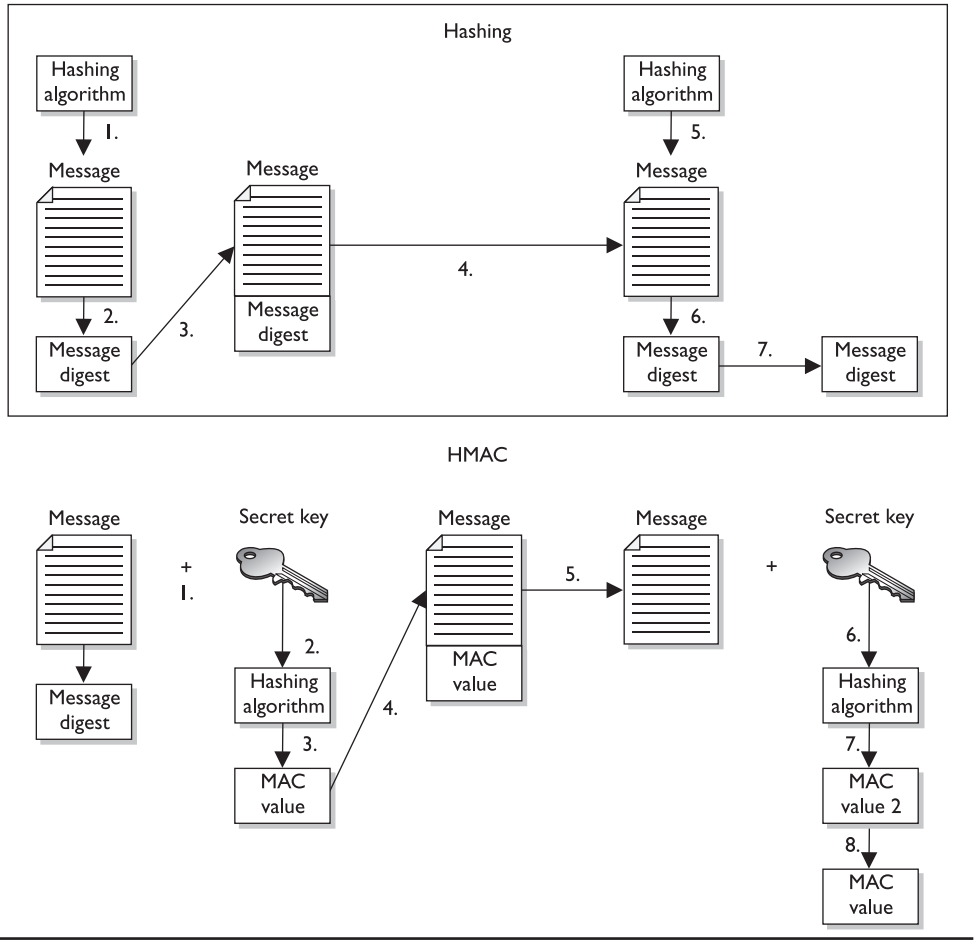


Figure 8-20 The steps involved in using a hashing algorithm and HMAC function

Why Can't We Call an Apple an Apple?

The idea of a hashing function is simple. You run a message through a hashing algorithm, which in turn generates a hashing value. It must have been too simple, because someone threw in a lot of terms to make it more confusing:

- A hashing value can also be called a ***message digest*** or ***fingerprint.***
- Hashing algorithms are also called ***nonkeyed message digests.***
- Two types of message authentication code (MAC) exist: hashed MAC (HMAC) and CBC-MAC.
- MAC is also sometimes called ***message integrity code (MIC)*** or ***modification detection code (MDC).***

Good luck, and may the force be with you.

The top portion of Figure 8-20 shows the steps of a hashing process:

1. The sender puts the message through a hashing function.
2. A message digest value is generated.
3. The message digest is appended to the message.
4. The sender sends the message to the receiver.
5. The receiver puts the message through a hashing function.
6. The receiver generates her own message digest value.
7. The receiver compares the two message digest values. If they are the same, the message has not been altered.

The bottom half of Figure 8-20 shows the steps of an HMAC:

1. The sender concatenates a symmetric key with the message.
2. The result is put through a hashing algorithm.
3. A MAC value is generated.
4. The MAC value is appended to the message.
5. The sender sends the message to the receiver. (Just the message with the attached MAC value. The sender does not send the symmetric key with the message.)
6. The receiver concatenates a symmetric key with the message.

7. The receiver puts the results through a hashing algorithm and generates her own MAC value.
8. The receiver compares the two MAC values. If they are the same, the message has not been modified.

Now, when we say that the message is concatenated with a symmetric key, we don't mean a symmetric key is used to encrypt the message. The message is not encrypted in an HMAC function, so there is no confidentiality being provided. Think about throwing a message in a bowl and then throwing a symmetric key in the same bowl. If you dump the contents of the bowl into a hashing algorithm, the result will be a MAC value.

This type of technology requires the sender and receiver to have the same symmetric key. The HMAC function does not involve getting the symmetric key to the destination securely. That would have to happen through one of the other technologies we have discussed already (Diffie-Hellman and key agreement, or RSA and key exchange).

CBC-MAC

If a CBC-MAC is being used, the message is encrypted with a symmetric block cipher in CBC mode and the output of the final block of ciphertext is used as the MAC. The sender does not send the encrypted version of the message, but instead sends the plaintext version and the MAC attached to the message. The receiver receives the plaintext message and encrypts it with the same symmetric block cipher in CBC mode and calculates an independent MAC value. The receiver compares the new MAC value with the MAC value sent with the message. This method does not use a hashing algorithm, as does HMAC.

The use of the symmetric key ensures that the only person who can verify the integrity of the message is the person who has a copy of this key. No one else can verify the data's integrity, and if someone were to make a change to the data, he could not generate the MAC value (HMAC or CBC-MAC) the receiver would be looking for. Any modifications would be detected by the receiver.

Now the receiver knows that the message came from the system that has the other copy of the same symmetric key, so MAC provides a form of authentication. It provides ***data origin authentication,*** sometimes referred to as ***system authentication.*** This is different from user authentication, which would require the use of a private key. A private key is bound to an individual; a symmetric key is not. MAC authentication provides the weakest form of authentication because it is not bound to a user, just to a computer or device.

NOTE The same key should not be used for authentication and encryption.

Hashes, HMACs, and CBC-MACs—Oh My!

MACs and hashing processes can be confusing. The following table simplifies the differences between them.

Function	Steps	Security Service Provided
Hash	1. Sender puts a message through a hashing algorithm and generates a message digest (MD) value. 2. Sender sends message and MD value to receiver. 3. Receiver runs just the message through the same hashing algorithm and creates an independent MD value. 4. Receiver compares both MD values. If they are the same, the message was not modified.	Integrity; not confidentiality or authentication. Can detect only unintentional modifications.
HMAC	1. Sender concatenates a message and secret key and puts the result through a hashing algorithm. This creates a MAC value. 2. Sender appends the MAC value to the message and sends it to the receiver. 3. The receiver takes just the message and concatenates it with her own symmetric key. This results in an independent MAC value. 4. The receiver compares the two MAC values. If they are the same, the receiver knows the message was not modified and knows from which system it came.	Integrity and data origin authentication; confidentiality is not provided.
CBC-MAC	1. Sender encrypts a message with a symmetric block algorithm in CBC mode. 2. The last block is used as the MAC. 3. The plaintext message and the appended MAC are sent to the receiver. 4. The receiver encrypts the message, creates a new MAC, and compares the two values. If they are the same, the receiver knows the message was not modified and from which system it came.	Data origin authentication and integrity.

Various Hashing Algorithms

As stated earlier, the goal of using a one-way hash function is to provide a fingerprint of the message. If two different messages produce the same hash value, it would be easier for an attacker to break that security mechanism, because patterns would be revealed.

A strong one-hash function should not provide the same hash value for two or more different messages. If a hashing algorithm takes steps to ensure it does not create the same hash value for two or more messages, it is said to be *collision free.*

Good cryptographic hash functions should have the following characteristics:

- The hash should be computed over the entire message.
- The hash should be a one-way function so messages are not disclosed by their values.
- Given a message and its hash value, computing another message with the same hash value should be impossible.
- The function should be resistant to birthday attacks (explained in the upcoming section "Attacks Against One-Way Hash Functions").

Table 8-2 and the following sections quickly describe some of the available hashing algorithms used in cryptography today.

MD2

MD2 is a one-way hash function designed by Ron Rivest that creates a 128-bit message digest value. It is not necessarily any weaker than the other algorithms in the "MD" family, but it is much slower.

MD4

MD4 is a one-way hash function designed by Ron Rivest. It also produces a 128-bit message digest value. It is used for high-speed computation in software implementations and is optimized for microprocessors.

MD5

MD5 was also created by Ron Rivest and is the newer version of MD4. It still produces a 128-bit hash, but the algorithm is more complex, which makes it harder to break.

Algorithm	Description
Message Digest 2 (MD2) algorithm	One-way function. Produces a 128-bit hash value. Much slower than MD4 and MD5.
Message Digest 4 (MD4) algorithm	One-way function. Produces a 128-bit hash value.
Message Digest 5 (MD5) algorithm	One-way function. Produces a 128-bit hash value. More complex than MD4.
HAVAL	One-way function. Variable-length hash value. Modification of MD5 algorithm that provides more protection against attacks that affect MD5.
Secure Hash Algorithm (SHA)	One-way function. Produces a 160-bit hash value. Used with DSA.
SHA-1, SHA-256, SHA-384, SHA-512	Updated version of SHA. SHA-1 produces a 160-bit hash value, SHA-256 creates a 256-bit value, and so on.

Table 8-2 Various Hashing Algorithms Available

MD5 added a fourth round of operations to be performed during the hashing functions and makes several of its mathematical operations carry out more steps or more complexity to provide a higher level of security.

SHA

SHA was designed by NIST and NSA to be used with the Digital Signature Standard (DSS). SHA was designed to be used in digital signatures and was developed when a more secure hashing algorithm was required for U.S. government applications.

SHA produces a 160-bit hash value, or message digest. This is then inputted into an asymmetric algorithm, which computes the signature for a message.

SHA is similar to MD4. It has some extra mathematical functions and produces a 160-bit hash instead of a 128-bit hash, which makes it more resistant to brute force attacks, including birthday attacks.

SHA was improved upon and renamed SHA-1. Recently, newer versions of this algorithm have been developed and released: SHA-256, SHA-384, and SHA-512.

HAVAL

HAVAL is a variable-length one-way hash function and is a modification of MD5. It processes message blocks twice the size of those used in MD5; thus, it processes blocks of 1024 bits.

Tiger

Ross Anderson and Eli Biham developed a hashing algorithm called Tiger in 1995. It was designed to carry out hashing functionalities on 64-bit systems and to be faster than MD5 and SHA-1. The resulting hash value is 192 bits. Design-wise most hash algorithms (MD5, RIPEMD, SHA0, and SHA1) are derivatives, or have been built upon the MD4 architecture. Tiger was built upon a different type of architecture with the goal of not being vulnerable to the same type of attacks that could be successful towards the other hashing algorithms. To fully understand how tested attacks have been carried out on the Tiger algorithm, you can view the document at http://th.informatik.uni-mannheim.de/People/Lucks/papers/Tiger_FSE_v10.pdf.

NOTE A European project called RIPE (RACE Integrity Primitives Evaluation) developed a hashing algorithm with the purpose of replacing MD4. This algorithm is called RIPEMD and is very similar to MD4, but did not gain much attention.

References

- **Counterpane Internet Security, Inc. home page** www.counterpane.com
- **The Search Directory: Cryptography, 3DES, Hash algorithm, MD5, SHA-1 links** www.the-search-directory.com/cryptography
- **RFC 1321 – The MD5 Message-Digest Algorithm** www.faqs.org/rfcs/rfc1321.html
- **"Asymmetric Cryptography," sample chapter from *.NET Security and Cryptography*, by Peter Thorsteinson and G. Ganesh (Prentice Hall PTR, August 2003)** www.informit.com/articles/article.asp?p=102212&seqNum=3

- **iNFOSYSSEC cryptography, encryption, and stenography links**
  www.infosyssec.net/infosyssec/cry2.htm

Attacks Against One-Way Hash Functions

A good hashing algorithm should not produce the same hash value for two different messages. If the algorithm does produce the same value for two distinctly different messages, this is called a ***collision.*** An attacker can attempt to force a collision, which is referred to as a ***birthday attack.*** This attack is based on the mathematical birthday paradox that exists in standard statistics. Now hold on to your hat while we go through this—it is a bit tricky:

How many people must be in the same room for the chance to be greater than even that another person has the same birthday as you?

Answer: 253

How many people must be in the same room for the chance to be greater than even that at least two people share the same birthday?

Answer: 23

This seems a bit backwards, but the difference is that in the first instance, you are looking for someone with a specific birthday date that matches yours. In the second instance, you are looking for any two people who share the same birthday. There is a higher probability of finding two people who share a birthday than of finding another person who shares your birthday. Or, stated another way, it is easier to find two matching values in a sea of values than to find a match for just one specific value.

Why do we care? The birthday paradox can apply to cryptography as well. Since any random set of 23 people most likely (at least a 50 percent chance) includes two people who share a birthday, by extension, if a hashing algorithm generates a message digest of 60 bits, there is a high likelihood that an adversary can find a collision using only 2^{30} inputs.

The main way an attacker can find the corresponding hashing value that matches a specific message is through a brute force attack. If he finds a message with a specific hash value, it is equivalent to finding someone with a specific birthday. If he finds two messages with the same hash values, it is equivalent to finding two people with the same birthday.

The output of a hashing algorithm is n, and to find a message through a brute force attack that results in a specific hash value would require hashing 2^n random messages. To take this one step further, finding two messages that hash to the same value would require review of only $2^{n/2}$ messages.

How Would a Birthday Attack Take Place?

Sue and Joe are going to get married, but before they do, they have a prenuptial contract drawn up that states if they get divorced, then Sue takes her original belongings and Joe takes his original belongings. To ensure this contract is not modified, it is hashed and a message digest value is created.

One month after Sue and Joe get married, Sue carries out some devious activity behind Joe's back. She makes a copy of the message digest value without anyone knowing. Then she makes a new contract that states that if Joe and Sue get a divorce, Sue

owns both her own original belongings and Joe's original belongings. Sue hashes this new contract and compares the new message digest value with the message digest value that correlates with the contract. They don't match. So Sue tweaks her contract ever so slightly and creates another message digest value and compares them. She continues to tweak her contract until she forces a collision, meaning her contract creates the same message digest value as the original contract. Sue then changes out the original contract with her new contract and quickly divorces Joe. When Sue goes to collect Joe's belongings and he objects, she shows him that no modification could have taken place on the original document because it still hashes out to the same message digest. Sue then moves to an island.

Hash algorithms usually use message digest sizes (the value of *n*) that are large enough to make collisions difficult to accomplish, but they are still possible. An algorithm that has 160-bit output, like SHA-1, may require approximately 2^{80} computations to break. This means there is a less than 1 in 2^{80} chance that someone could carry out a successful birthday attack.

The main point of discussing this paradox is to show how important longer hashing values truly are. A hashing algorithm that has a larger bit output is less vulnerable to brute force attacks such as a birthday attack. This is the primary reason why the new versions of SHA have such large message digest values.

Reference

- ***The PGP Attack FAQ*, Part 3, "The One-Way Hash"** www.stack.nl/~galactus/remailers/attack-3.html

Digital Signatures

To do a digital signature, do I sign my name on my monitor screen?
Response: Sure.

A digital signature is a hash value that has been encrypted with the sender's private key. The act of signing means encrypting the message's hash value with a private key, as shown in Figure 8-21.

From our earlier example in the "One-Way Hash" section, if Kevin wants to ensure that the message he sends to Maureen is not modified *and* he wants her to be sure it came only from him, he can digitally sign the message. This means that a one-way hashing function would be run on the message, and then Kevin would encrypt that hash value with his private key.

When Maureen receives the message, she will perform the hashing function on the message and come up with her own hash value. Then she will decrypt the sent hash value (digital signature) with Kevin's public key. She then compares the two values, and if they are the same, she can be sure the message was not altered during transmission. She is also sure the message came from Kevin because the value was encrypted with his private key.

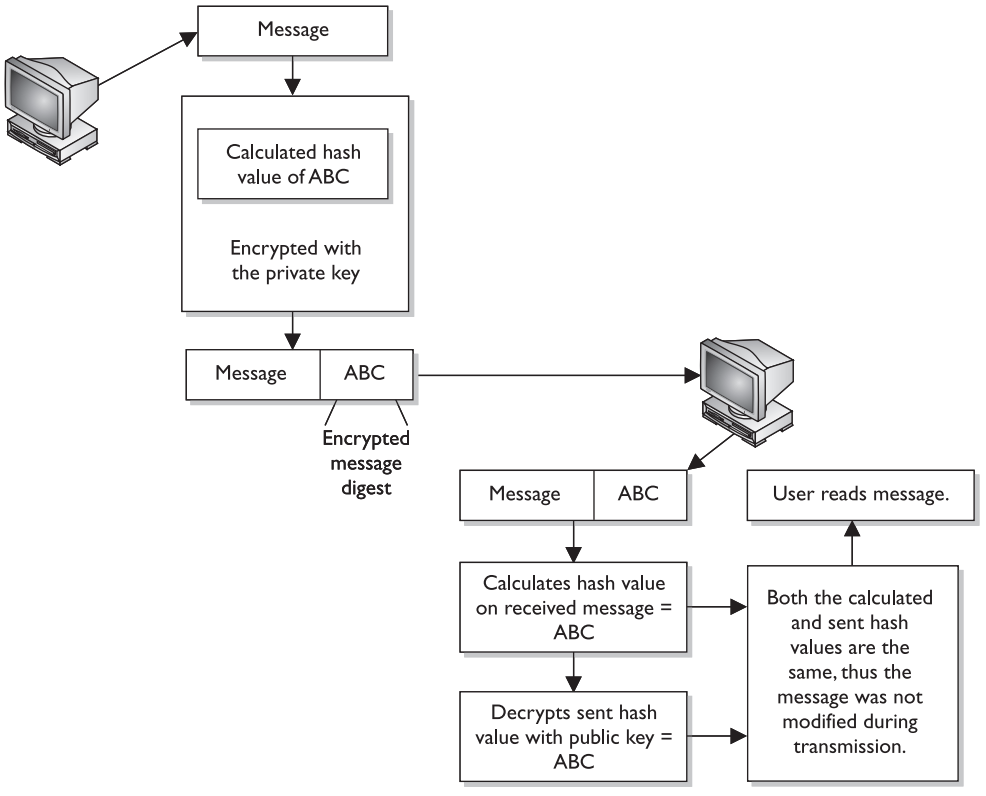


Figure 8-21 Creating a digital signature for a message

The hashing function ensures the integrity of the message, and the signing of the hash value provides authentication and nonrepudiation. The act of signing just means the value was encrypted with a private key.

We need to be clear on all the available choices within cryptography, because different steps and algorithms provide different types of security services:

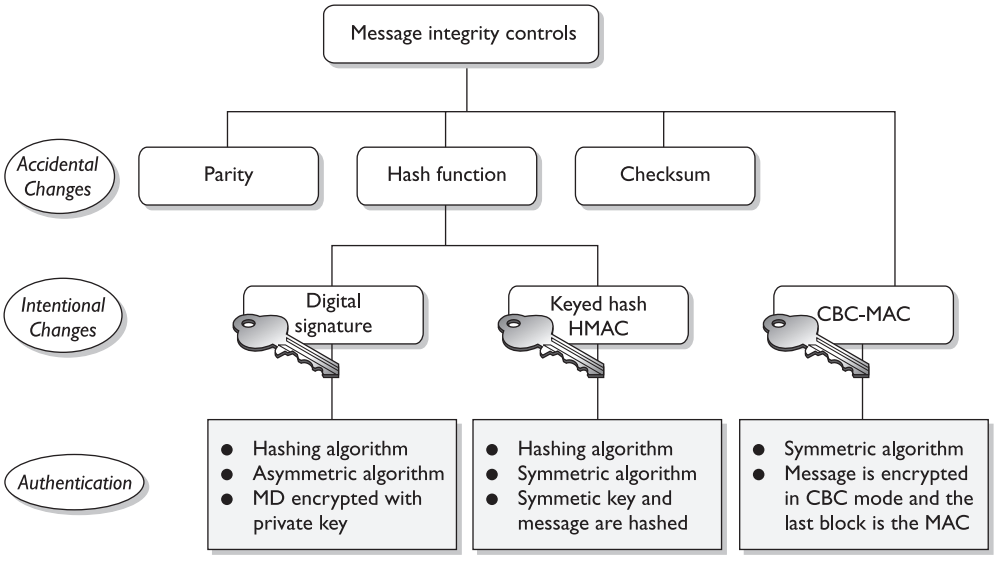
- A message can be encrypted, which provides confidentiality.
- A message can be hashed, which provides integrity.
- A message can be digitally signed, which provides authentication, nonrepudiation, and integrity.
- A message can be encrypted and digitally signed, which provides confidentiality, authentication, nonrepudiation, and integrity.

Some algorithms can only perform encryption, whereas others support digital signatures and encryption. When hashing is involved, a hashing algorithm is used, not an encryption algorithm.

It is important to understand that not all algorithms can necessarily provide all security services. Most of these algorithms are used in some type of combination to provide all the necessary security services required of an environment. Table 8-3 shows the services provided by the algorithms.

Algorithm Type	Encryption	Digital Signature	Hashing Function	Key Distribution
Asymmetric Key Algorithms				
RSA	X	X		X
ECC	X	X		X
Diffie-Hellman				X
El Gamal	X	X		X
DSA		X		
LUC	X	X		X
Knapsack	X	X		X
Symmetric Key Algorithms				
DES	X			
3DES	X			
Blowfish	X			
IDEA	X			
RC4	X			
SAFER	X			
Hashing Algorithms				
Ronald Rivest family of hashing functions: MD2, MD4, and MD5			X	
SHA			X	
HAVAL (variable-length hash values using a one-way function design)			X	

Table 8-3 Various Functions of Different Algorithms



Digital Signature Standard

Because digital signatures are so important in proving who sent which messages, the U.S. government decided to establish standards pertaining to their functions and acceptable use. In 1991, NIST proposed a federal standard called the ***Digital Signature Standard (DSS).*** It was developed for federal departments and agencies, but most vendors designed their products to meet these specifications also. The federal government requires its departments to use DSA, RSA, or the elliptic curve digital signature algorithm (ECDSA) and SHA. SHA creates a 160-bit message digest output, which is then inputted into one of the three mentioned digital signature algorithms. SHA is used to ensure the integrity of the message, and the other algorithms are used to digitally sign the message. This is an example of how two different algorithms are combined to provide the right combination of security services.

RSA and DSA are the best known and most widely used digital signature algorithms. DSA was developed by the NSA. Unlike RSA, DSA can be used only for digital signatures, and DSA is slower than RSA in signature verification. RSA can be used for digital signatures, encryption, *and* secure distribution of symmetric keys.

Public Key Infrastructure

Let's put all of these cryptography pieces in a bowl and figure out how they all work together.

Public key infrastructure (PKI) consists of programs, data formats, procedures, communication protocols, security policies, and public key cryptographic mechanisms working in a comprehensive manner to enable a wide range of dispersed people to

communicate in a secure and predictable fashion. In other words, a PKI establishes a level of trust within an environment. PKI is an ISO authentication framework that uses public key cryptography and the X.509 standard. The framework was set up to enable authentication to happen across different networks and the Internet. Particular protocols and algorithms are not specified, which is why PKI is called a framework and not a specific technology.

PKI provides authentication, confidentiality, nonrepudiation, and integrity of the messages exchanged. PKI is a *hybrid* system of symmetric and asymmetric key algorithms and methods, which were discussed in earlier sections.

There is a difference between public key cryptography and PKI. Public key cryptography is another name for asymmetric algorithms, while PKI is what its name states—it is an infrastructure. The infrastructure assumes that the receiver's identity can be positively ensured through certificates and that an asymmetric algorithm will automatically carry out the process of key exchange. The infrastructure therefore contains the pieces that will identify users, create and distribute certificates, maintain and revoke certificates, distribute and maintain encryption keys, and enable all technologies to communicate and work together for the purpose of encrypted communication and authentication.

Public key cryptography is one piece in PKI, but many other pieces make up this infrastructure. An analogy can be drawn with the e-mail protocol Simple Mail Transfer Protocol (SMTP). SMTP is the technology used to get e-mail messages from here to there, but many other things must be in place before this protocol can be productive. We need e-mail clients, e-mail servers, and e-mail messages, which together build a type of infrastructure—an e-mail infrastructure. PKI is made up of many different parts: certificate authorities, registration authorities, certificates, keys, and users. The following sections explain these parts and how they all work together.

Each person who wants to participate in a PKI requires a digital certificate, which is a credential that contains the public key for that individual along with other identifying information. The certificate is created and signed (digital signature) by a trusted third party, which is a certificate authority (CA). When the CA signs the certificate, it binds the individual's identity to the public key, and the CA takes liability for the authenticity of that individual. It is this trusted third party (the CA) that allows people who have never met to authenticate to each other and communicate in a secure method. If Kevin has never met David, but would like to communicate securely with him, and they both trust the same CA, then Kevin could retrieve David's digital certificate and start the process.

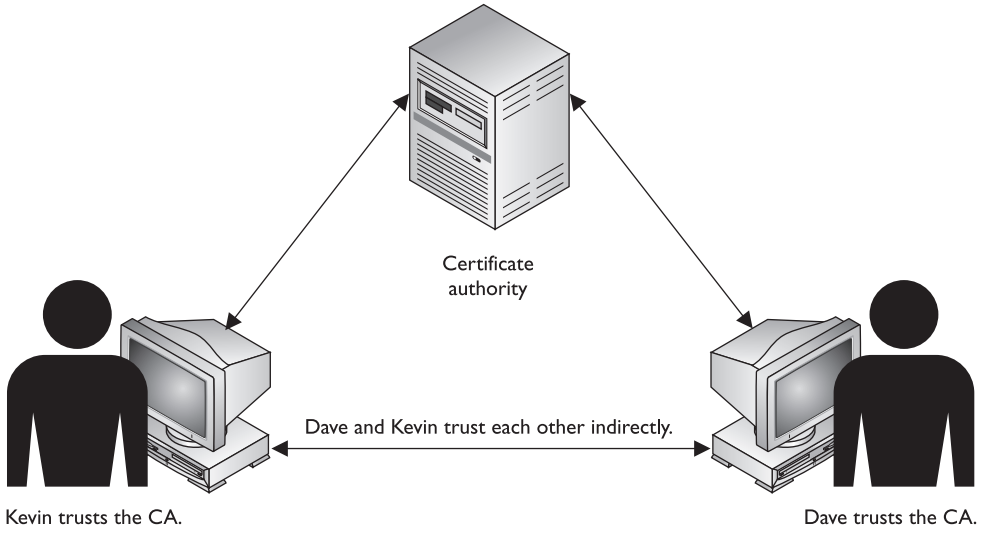
Certificate Authorities

How do I know I can trust you?
Response: The CA trusts me.

A CA is a trusted organization (or server) that maintains and issues digital certificates. When a person requests a certificate, the registration authority (RA) verifies that individual's identity and passes the certificate request off to the CA. The CA constructs the certificate, signs it, sends it to the requester, and maintains the certificate over its lifetime. When another person wants to communicate with this person, the CA will basically vouch for that person's identity. When David receives a digital certificate from Kevin, David will go through steps to validate it. Basically, by providing David with his digital certificate, Kevin is stating, "I know you don't know or trust me, but here is this

document that was created by someone you do know and trust. The document says I am a good guy and you should trust me."

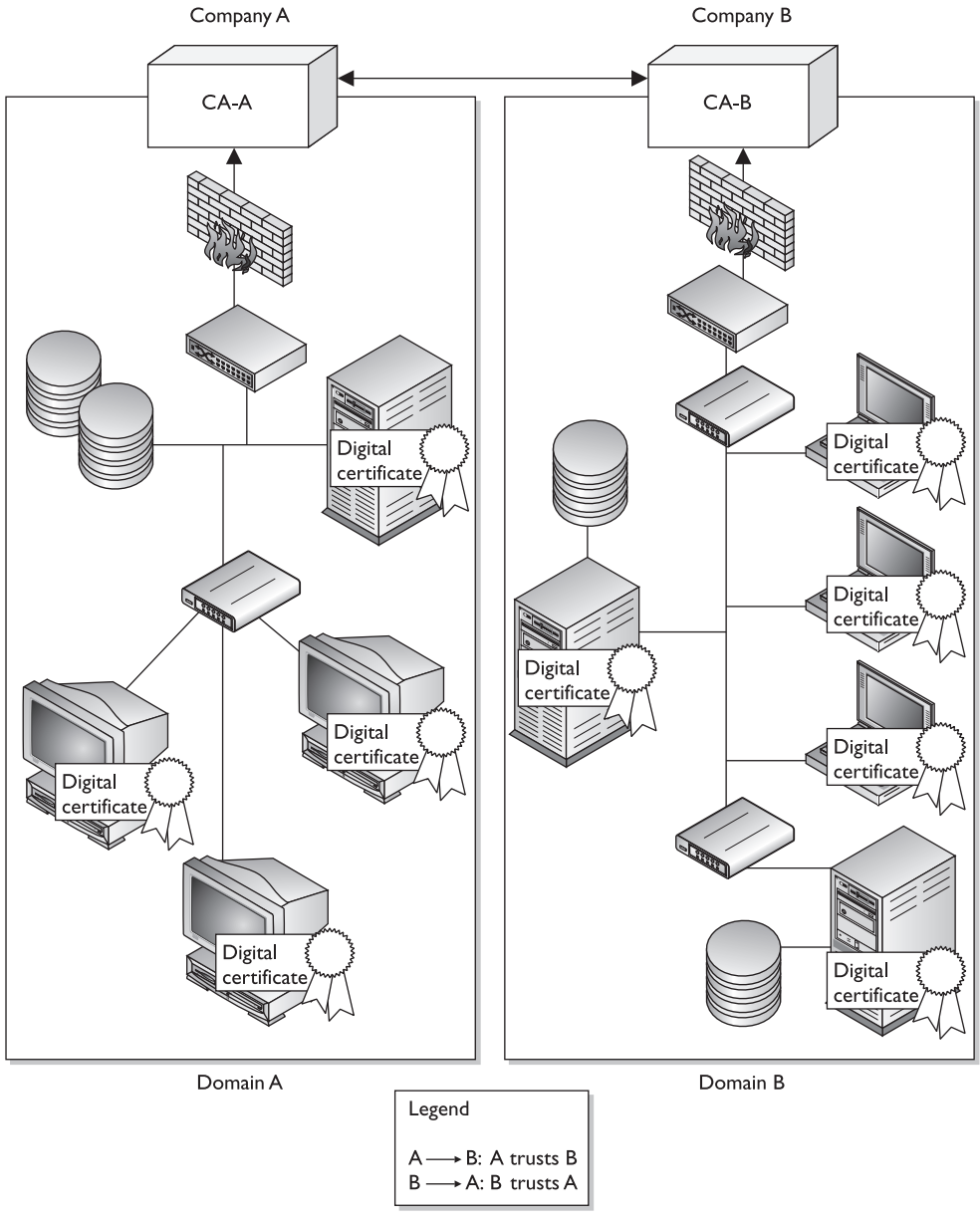
Once David validates the digital certificate, he extracts Kevin's public key, which is embedded within it. Now David knows this public key is bound to Kevin. He also knows that if Kevin uses his private key to create a digital signature and David can properly decrypt it using this public key, it did indeed come from Kevin.



NOTE Remember the man-in-the-middle attack covered earlier in the "Diffie-Hellman Algorithm" section? This attack is possible if two users are not working in a PKI environment and do not truly know the identity of the owners of the public keys.

The CA can be internal to an organization. Such a setup would enable the company to control the CA server, configure how authentication takes place, maintain the certificates, and recall certificates when necessary. Other CAs are organizations dedicated to this type of service, and other individuals and companies pay them to supply it. Some well-known CAs are Entrust and VeriSign. Many browsers have several well-known CAs configured by default.

NOTE More and more organizations are setting up their own internal PKIs. When these independent PKIs need to interconnect to allow for secure communication to take place (either between departments or different companies), there must be a way for the two root CAs to trust each other. The two CAs do not have a CA above them they can both trust, so they must carry out cross certification. A cross certification is the process undertaken by CAs to establish a trust relationship in which they rely upon each other's digital certificates and public keys as if they had issued them themselves. When this is set up, a CA for one company can validate digital certificates from the other company and vice versa.



The CA is responsible for creating and handing out certificates, maintaining them, and revoking them if necessary. Revocation is handled by the CA and the revoked certificate information is stored on a ***certificate revocation list (CRL)***. This is a list of every certificate that has been revoked. This list is maintained and updated periodically. A certificate may be revoked because the key holder's private key was compromised or because the CA discovered the certificate was issued to the wrong person. An analogy for the use of a CRL is how a driver's license is used by a police officer. If an officer pulls

over Sean for speeding, the officer will ask to see Sean's license. The officer will then run a check on the license to find out if Sean is wanted for any other infractions of the law and to verify the license has not expired. The same thing happens when a person compares a certificate to a CRL. If the certificate became invalid for some reason, the CRL is the mechanism for the CA to let others know this information.

NOTE CRLs are the thorn in the side of many PKI implementations. They are challenging for a long list of reasons. It is interesting to know that, by default, web browsers do not check a CRL to ensure that a certificate is not revoked. So when you are setting up an SSL connection to do e-commerce over the Internet, you could be relying on a certificate that has actually been revoked. Not good.

Online Certificate Status Protocol (OCSP) is being used more and more compared to the cumbersome CRL approach. When using just a CRL, the user's browser must either check a central CRL to find out if the certification has been revoked or continually push out CRL values to the clients to ensure they have an updated CRL. If OCSP is implemented, it does this work automatically in the background. It carries out real-time validation of a certificate and reports back to the user whether the certificate is valid, invalid, or unknown. OCSP checks the CRL that is maintained by the CA. So the CRL is still being used, but now we have a protocol developed specifically to check the CRL during a certificate validation process.

Certificates

One of the most important pieces of a PKI is its digital certificate. A ***certificate*** is the mechanism used to associate a public key with a collection of components in a manner that is sufficient to uniquely identify the claimed owner. The standard for how the CA creates the certificate is X.509, which dictates the different fields used in the certificate and the valid values that can populate those fields. We are currently at version 4 of this standard, which is often denoted as X.509v4. Many cryptographic protocols use this type of certificate, including SSL.

The certificate includes the serial number, version number, identity information, algorithm information, lifetime dates, and the signature of the issuing authority, as shown in Figure 8-22.

The Registration Authority

The ***registration authority (RA)*** performs the certification registration duties. The RA establishes and confirms the identity of an individual, initiates the certification process with a CA on behalf of an end user, and performs certificate life-cycle management functions. The RA cannot issue certificates, but can act as a broker between the user and the CA. When users need new certificates, they make requests to the RA, and the RA verifies all necessary identification information before allowing a request to go to the CA.

NOTE A few years ago, VeriSign assigned some very powerful certificates to two individuals who claimed to work for Microsoft. They did not work for Microsoft and this opened the door to a lot of potential security compromises.

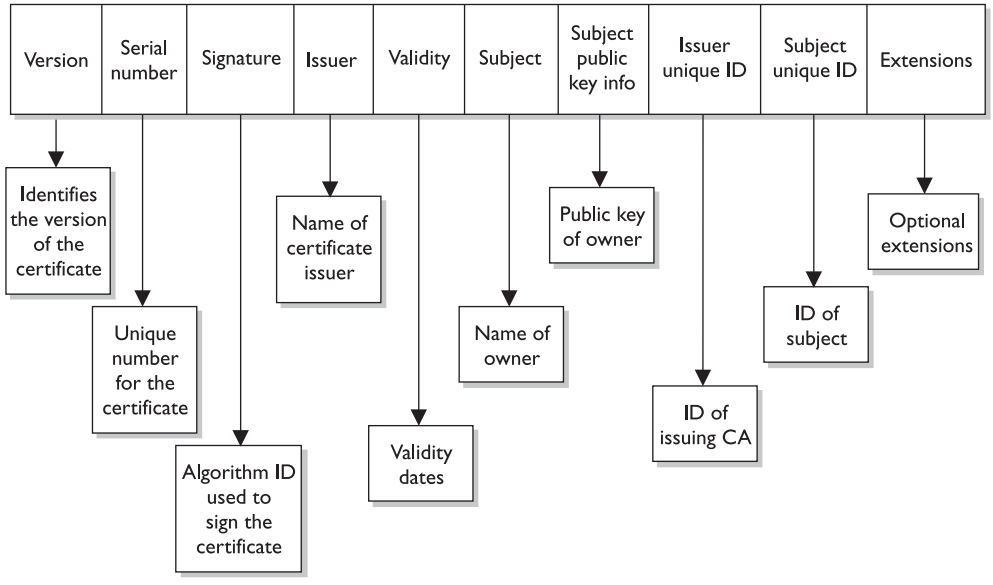


Figure 8-22 Each certificate has a structure with all the necessary identifying information in it.

PKI Steps

Now that we know some of the main pieces of a PKI and how they actually work together, let's walk through an example. First, suppose that John needs to obtain a digital certificate for himself so he can participate in a PKI. The following are the steps to do so:

1. John makes a request to the RA.
2. The RA requests certain identification information from John, such as a copy of his driver's license, his phone number, his address, and other identifying information.
3. Once the RA receives the required information from John and verifies it, the RA sends his certificate request to the CA.
4. The CA creates a certificate with John's public key and identity information embedded. (The private/public key pair is generated either by the CA or on John's machine, which depends on the systems' configurations. If it is created at the CA, his private key needs to be sent to him by secure means. In most cases, the user generates this pair and sends in his public key during the registration process.)

   Now John is registered and can participate in a PKI. John and Diane decide they want to communicate, so they take the following steps, shown in Figure 8-23.
5. John requests Diane's public key from a public directory.
6. The directory, sometimes called a repository, sends Diane's digital certificate.
7. John verifies the digital certificate and extracts her public key. John uses this public key to encrypt a session key that will be used to encrypt their messages.

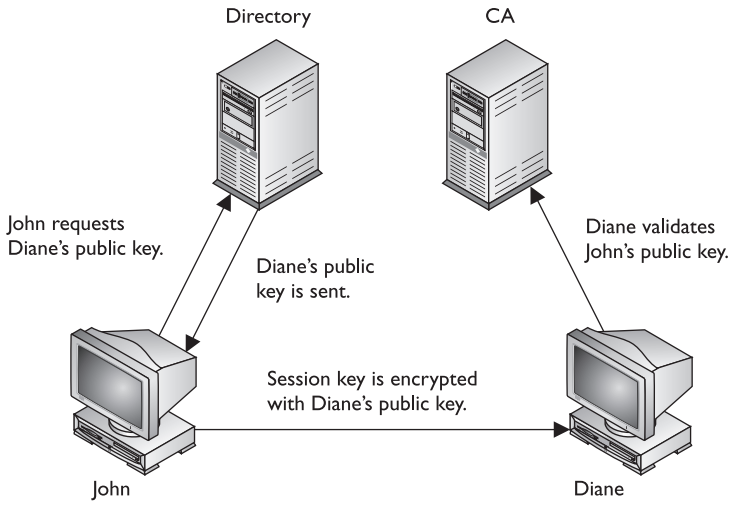


Figure 8-23 CA and user relationships

John sends the encrypted session key to Diane. John also sends his certificate, containing his public key, to Diane.

8. When Diane receives John's certificate, her browser looks to see if it trusts the CA that digitally signed this certificate. Diane's browser trusts this CA and, after she verifies the certificate, both John and Diane can communicate using encryption.

A PKI may be made up of the following entities and functions:

- CA
- RA
- Certificate repository
- Certificate revocation system
- Key backup and recovery system
- Automatic key update
- Management of key histories
- Timestamping
- Client-side software

PKI supplies the following security services:

- Confidentiality
- Access control
- Integrity
- Authentication
- Nonrepudiation

A PKI must retain a key history, which keeps track of all the old and current public keys that have been used by individual users. For example, if Kevin encrypted a symmetric key with David's old public key, there should be a way for David to still access this data. This can only happen if the CA keeps a proper history of David's old certificates and keys.

NOTE Another important component that must be integrated into a PKI is a reliable time source that provides a way for secure timestamping. This comes into play when true nonrepudiation is required.

References

- **Network World Cryptography page** www.nwfusion.com/research/crypto.html
- ***Introduction to Public Key Technology and the Federal PKI Infrastructure*, by D. Richard Kuhn *et al.*, NIST Special Publication 800-32 (Feb. 26, 2001)** http://csrc.nist.gov/publications/nistpubs/800-32/sp800-32.pdf
- **The Open Group Secure Messaging Toolkit** www.opengroup.org/messaging/G260/pki_tutorial.htm

Key Management

I am the manager of all keys!
Response: I feel so sorry for you.

Cryptography can be used as a security mechanism to provide confidentiality, integrity, and authentication, but not if the keys are compromised in any way. The keys can be captured, modified, corrupted, or disclosed to unauthorized individuals. Cryptography is based on a trust model. Individuals must trust each other to protect their own keys, trust the administrator who is maintaining the keys, and trust a server that holds, maintains, and distributes the keys.

Many administrators know that key management causes one of the biggest headaches in cryptographic implementation. There is more to key maintenance than using them to encrypt messages. The keys must be distributed securely to the right entities and updated continuously. They must also be protected as they are being transmitted and while they are being stored on each workstation and server. The keys must be generated, destroyed, and recovered properly. Key management can be handled through manual or automatic processes.

The keys are stored before and after distribution. When a key is distributed to a user, it does not just hang out on the desktop. It needs a secure place within the file system to be stored and used in a controlled method. The key, the algorithm that will use the key, configurations, and parameters are stored in a module that also needs to be protected. If an attacker is able to obtain these components, she could masquerade as another user and decrypt, read, and re-encrypt messages not intended for her.

Historically, physical cryptographic keys were kept in secured boxes and delivered by escorted couriers. The keys could be distributed to a main server, and then the local administration would distribute them, or the courier would visit each computer individually. Some implementations distributed a master key to a site, and then that key was

used to generate unique secret keys to be used by individuals at that location. Today, most key distributions are handled by a protocol through automated means and not manually by an individual. A company must evaluate the overhead of key management, the required security level, and cost-benefit issues to decide on how it will conduct key management, but overall, automation provides a more accurate and secure approach.

When using the Kerberos protocol (described in Chapter 4), a Key Distribution Center (KDC) is used to store, distribute, and maintain cryptographic session and secret keys. This method provides an automated method of key distribution. The computer that wants to access a service on another computer requests access via the KDC. The KDC then generates a session key to be used between the requesting computer and the computer providing the requested resource or service. The automation of this process reduces the possible errors that can happen through a manual process, but if the ticket granting service (TGS) portion of the KDC gets compromised in any way, then all the computers and their services are affected and possibly compromised.

In some instances, keys are still managed through manual means. Unfortunately, although many companies use cryptographic keys, they rarely if ever change them, either because of the hassle of key management or because the network administrator is already overtaxed with other tasks or does not realize the task actually needs to take place. The frequency of use of a cryptographic key has a direct correlation to how often the key should be changed. The more a key is used, the more likely it is to be captured and compromised. If a key is used infrequently, then this risk drops dramatically. The necessary level of security and the frequency of use can dictate the frequency of key updates. A mom-and-pop diner might only change its cryptography keys every month, whereas an information warfare military unit might change them every day or every week. The important thing is to change the keys using a secure method.

Key management is the most challenging part of cryptography and also the most crucial. It is one thing to develop a very complicated and complex algorithm and key method, but if the keys are not securely stored and transmitted, it does not really matter how strong the algorithm is.

Key Management Principles

Keys should not be in cleartext outside the cryptography device. As stated previously, many cryptography algorithms are known publicly, which puts more stress on protecting the secrecy of the key. If attackers know how the actual algorithm works, in many cases all they need to figure out is the key to compromise a system. This is why keys should not be available in cleartext—the key is what brings secrecy to encryption.

These steps, and all of key distribution and maintenance, should be automated and hidden from the user. These processes should be integrated into software or the operating system. It only adds complexity and opens the doors for more errors when processes are done manually and depend upon end users to perform certain functions.

Keys are at risk of being lost, destroyed, or corrupted. Backup copies should be available and easily accessible when required. If data are encrypted and then the user accidentally loses the necessary key to decrypt it, this information would be lost forever if there were not a backup key to save the day. The application being used for cryptography may have key recovery options, or it may require copies of the keys to be kept in a secure place.

Different scenarios highlight the need for key recovery or backup copies of keys. For example, if Bob has possession of all the critical bid calculations, stock value information, and corporate trend analysis needed for tomorrow's senior executive presentation, and Bob has an unfortunate confrontation with a bus, someone is going to need to access this data after the funeral. As another example, if an employee leaves the company and has encrypted important documents on her computer before departing, the company would probably still want to access that data later. Similarly, if the vice president did not know that running a large magnet over the floppy disk that holds his private key was not a good idea, he would want his key replaced immediately instead of listening to a lecture about electromagnetic fields and how they rewrite sectors on media.

Of course, having more than one key increases the chance of disclosure, so a company needs to decide whether it wants to have key backups and, if so, what precautions to put into place to protect them properly. A company can choose to have multiparty control for emergency key recovery. This means that if a key must be recovered, more than one person is needed for this process. The key recovery process could require two or more other individuals to present their private keys or authentication information. These individuals should not all be members of the IT department. There should be a member from management, an individual from security, and one individual from the IT department. All of these requirements reduce the potential for abuse and would require collusion for fraudulent activities to take place.

Rules for Keys and Key Management

Key management is critical for proper protection. The following are responsibilities that fall under the key management umbrella:

- The key length should be long enough to provide the necessary level of protection.
- Keys should be stored and transmitted by secure means.
- Keys should be extremely random and the algorithm should use the full spectrum of the keyspace.
- The key's lifetime should correspond with the sensitivity of the data it is protecting. (Less secure data may allow for a longer key lifetime, whereas more sensitive data might require a shorter key lifetime.)
- The more the key is used, the shorter its lifetime should be.
- Keys should be backed up or escrowed in case of emergencies.
- Keys should be properly destroyed when their lifetime comes to an end.

References

- **QuaMeta Compendium of Electronic Commerce Resources: Cryptography** www.quameta.com/security.htm
- **SSH Communications Security Support: Cryptography A–Z** www.ssh.fi/tech/crypto/intro.html

Link Encryption vs. End-to-End Encryption

Encryption can be performed at different communication levels, each with different types of protection and implications. Two general modes of encryption implementation are link encryption and end-to-end encryption. ***Link encryption*** encrypts all the data along a specific communication path, as in a satellite link, T3 line, or telephone circuit. Not only is the user information encrypted, but the header, trailers, addresses, and routing data that are part of the packets are also encrypted. The only traffic not encrypted in this technology is the data link control messaging information, which includes instructions and parameters that the different link devices use to synchronize communication methods. Link encryption provides protection against packet sniffers and eavesdroppers. In ***end-to-end encryption,*** the headers, addresses, routing, and trailer information are not encrypted, enabling attackers to learn more about a captured packet and where it is headed.

Link encryption, which is sometimes called *online encryption,* is usually provided by service providers and is incorporated into network protocols. All of the information is encrypted, and the packets must be decrypted at each hop so the router, or other intermediate device, knows where to send the packet next. The router must decrypt the header portion of the packet, read the routing and address information within the header, and then re-encrypt it and send it on its way.

With end-to-end encryption, the packets do not need to be decrypted and then encrypted again at each hop, because the headers and trailers are not encrypted. The devices in between the origin and destination just read the necessary routing information and pass the packets on their way.

End-to-end encryption is usually initiated by the user of the originating computer. It provides more flexibility for the user to be able to determine whether or not certain messages will get encrypted. It is called "end-to-end encryption" because the message stays encrypted from one end of its journey to the other. Link encryption has to decrypt the packets at every device between the two ends.

Link encryption occurs at the data link and physical layers, as depicted in Figure 8-24. Hardware encryption devices interface with the physical layer and encrypt all data that pass through them. Because no part of the data is available to an attacker, the attacker cannot learn basic information about how data flows through the environment. This is referred to as *traffic-flow security.*

Encryption at Different Layers

In reality, encryption can happen at different layers of an operating system and network stack. The following are just a few examples:

- End-to-end encryption happens within the applications.
- SSL encryption takes place at the transport layer.
- PPTP encryption takes place at the data link layer.
- Link encryption takes place at the data link and physical layers.

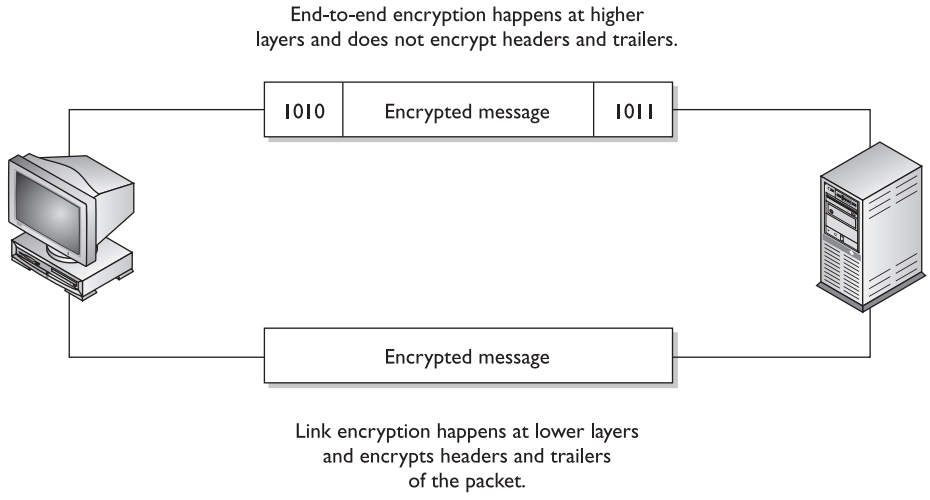


Figure 8-24 Link and end-to-end encryption happen at different OSI layers.

NOTE A *hop* is a device that helps a packet reach its destination. It is usually a router that looks at the packet address to determine where the packet needs to go next. Packets usually go through many hops between the sending and receiving computers.

The following list outlines the advantages and disadvantages of end-to-end and link encryption methods.

Advantages of end-to-end encryption include the following:

- It provides more flexibility to the user in choosing what gets encrypted and how.
- Higher granularity of functionality is available because each application or user can choose specific configurations.
- Each hop computer on the network does not need to have a key to decrypt each packet.

Disadvantages of end-to-end encryption include the following:

- Headers, addresses, and routing information are not encrypted, and therefore not protected.

Advantages of link encryption include the following:

- All data are encrypted, including headers, addresses, and routing information.
- Users do not need to do anything to initiate it. It works at a lower layer in the OSI model.

Disadvantages of link encryption include the following:

- Key distribution and management are more complex because each hop device must receive a key, and when the keys change, each must be updated.
- Packets are decrypted at each hop; thus, more points of vulnerability exist.

Hardware vs. Software Cryptography Systems

Encryption can be done through software or hardware, and there are trade-offs with each. Generally, software is less expensive and provides a slower throughput than hardware mechanisms. Software cryptography methods can be more easily modified and disabled compared to hardware systems, but it depends on the application and the hardware product.

If a company needs to perform high-end encryption functions at a higher speed, the company will most likely implement a hardware solution.

Reference

- ***On Distributed Communications: IX Security, Secrecy, and Tamper-Free Considerations*, Section 3, "Some Fundamentals of Cryptography," by Paul Baran (Rand Corp., August 1964)** www.rand.org/publications/RM/RM3765/RM3765.chapter3.html

E-mail Standards

Like other types of technologies, cryptography has industry standards and de facto standards. Standards are necessary because they help ensure interoperability among vendor products. Standards usually mean that a certain technology has been under heavy scrutiny and properly tested and accepted by many similar technology communities. A company still needs to decide on what type of standard to follow and what type of technology to implement.

A company needs to evaluate the functionality of the technology and perform a cost-benefit analysis on the competing products within the chosen standards. For a cryptography implementation, the company would need to decide what must be protected by encryption, whether digital signatures are necessary, how key management should take place, what type of resources are available to implement and maintain the technology, and what the overall cost will amount to.

If a company only needs to encrypt some e-mail messages here and there, then PGP may be the best choice. If the company wants all data encrypted as it goes throughout the network and to sister companies, then a link encryption implementation may be the best choice. If a company wants to implement a single sign-on environment where users need to authenticate to use different services and functionality throughout the network, then implementing a PKI or Kerberos might serve it best. The network administrators should understand each type of technology and standard, to help make the most informed decision, and should research and test each competing product within the chosen technology before making the final purchase. Cryptography can be a complicated subject, including how to implement and maintain it. Doing homework versus buying into buzzwords and flashy products might help a company reduce its headaches down the road.

The following sections briefly describe some of the most popular e-mail standards in use.

Multipurpose Internet Mail Extension

Multipurpose Internet Mail Extension (MIME) is a technical specification indicating how multimedia data and e-mail attachments are to be transferred. The Internet has mail standards that dictate how mail is to be formatted, encapsulated, transmitted, and opened. If a message or document contains a binary attachment, MIME dictates how that portion of the message should be handled.

When an attachment contains an audio clip, graphic, or some other type of multimedia component, the e-mail client will send the file with a header that describes the file type. For example, the header might indicate that the MIME type is Image and the subtype is jpeg. Although this will be in the header, many times systems also use the file's extension to identify the MIME type. So, in the preceding example, the file's name might be stuff.jpeg. The user's system will see the extension jpeg, or see the data in the header field, and look in its association list to see what program it needs to initialize to open this particular file. If the system has JPEG files associated with the Explorer application, then Explorer will open and present the picture to the user.

Sometimes systems either do not have an association for a specific file type or do not have the helper program necessary to review and use the contents of the file. When a file has an unassociated icon assigned to it, it might require the user to choose the Open With command and choose an application in the list to associate this file with that program. So when the user double-clicks that file, the associated program will initialize and present the file. If the system does not have the necessary program, the web site might offer the necessary helper program, like Acrobat or an audio program that plays WAV files.

MIME is a specification that dictates how certain file types should be transmitted and handled. This specification has several types and subtypes, enables different computers to exchange data in varying formats, and provides a standardized way of presenting the data. So if Sean views a funny picture that is in GIF format, he can be sure that when he sends it to Debbie, it will look exactly the same.

Secure MIME (S/MIME) is a standard for encrypting and digitally signing electronic mail and for providing secure data transmissions. S/MIME extends the MIME standard by allowing for the encryption of e-mail and attachments. The encryption and hashing algorithms can be specified by the user of the mail package, instead of having it dictated to them.

S/MIME provides confidentiality through the user's encryption algorithm, integrity through the user's hashing algorithm, authentication through the use of X.509 public key certificates, and nonrepudiation through cryptographically signed message digests.

Privacy-Enhanced Mail

Privacy-Enhanced Mail (PEM) is an Internet standard to provide secure e-mail over the Internet and for in-house communication infrastructures. The protocols within PEM provide authentication, message integrity, encryption, and key management. This standard was developed to provide compatibility with many types of key-management processes and symmetric and public key methods of encryption.

PEM is a series of message authentication and encryption technologies developed by several governing groups. PEM can use AES for encryption and RSA for sender au-

thentication and key management. It also provides support for nonrepudiation. The following are specific components that can be used in PEM:

- Messages encrypted with AES in CBC mode
- Public key management, provided by using RSA
- X.509 standard, used for certification structure and format

PEM has not caught on to the extent the developers had planned. The main issue is that PEM provides too much structure for different environments that require more flexibility in their secure communication infrastructure.

Message Security Protocol

The military has its own way of securing messages.
Response: It has its own way of doing most things.

The ***Message Security Protocol (MSP)*** is the military's PEM. Developed by the NSA, it is an X.400-compatible application-level protocol used to secure e-mail messages. MSP can be used to sign and encrypt messages and perform hashing functions. Like PEM, applications that incorporate MSP enable different algorithms and parameters to be used to provide greater flexibility.

References

- **Encryption and Security Tutorial, by Peter Gutmann** www.cs.auckland.ac.nz/~pgut001/tutorial
- ***Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML*, by Don Davis** http://world.std.com/~dtd/sign_encrypt/sign_encrypt7.html

Pretty Good Privacy

Pretty Good Privacy (PGP) was designed by Phil Zimmerman as a freeware e-mail security program and was released in 1991. It was the first widespread public key encryption program. PGP is a complete cryptosystem that uses cryptographic protection to protect e-mail and files. It can use RSA public key encryption for key management and use IDEA symmetric cipher for bulk encryption of data, although the user has the option of picking different types of algorithms for these functions. PGP can provide confidentiality by using the IDEA encryption algorithm, integrity by using the MD5 hashing algorithm, authentication by using the public key certificates, and nonrepudiation by using cryptographically signed messages. PGP uses its own type of digital certificates rather than what is used in PKI, but they both have similar purposes.

The user's private key is generated and encrypted when the application asks the user to randomly type on her keyboard for a specific amount of time. Instead of using passwords, PGP uses passphrases. The passphrase is used to encrypt the user's private key that is stored on her hard drive.

PGP does not use a hierarchy of CAs, or any type of formal trust certificates, but instead relies on a "web of trust" in its key management approach. Each user generates and

distributes his or her public key, and users sign each other's public keys, which creates a community of users who trust each other. This is different from the CA approach, where no one trusts each other; they only trust the CA. For example, if Mark and Joe want to communicate using PGP, Mark can give his public key to Joe. Joe signs Mark's key and keeps a copy for himself. Then, Joe gives a copy of his public key to Mark so they can start communicating securely. Later, Mark would like to communicate with Sally, but Sally does not know Mark and does not know if she can trust him. Mark sends Sally his public key, which has been signed by Joe. Sally has Joe's public key, because they have communicated before, and she trusts Joe. Because Joe signed Mark's public key, Sally now also trusts Mark and sends her public key and begins communicating with him.

So, basically, PGP is a system of "I don't know you, but my buddy Joe says you are an all right guy, so I will trust you on Joe's word."

Each user keeps in a file, referred to as a ***key ring,*** a collection of public keys he has received from other users. Each key in that ring has a parameter that indicates the level of trust assigned to that user and the validity of that particular key. If Steve has known Liz for many years and trusts her, he might have a higher level of trust indicated on her stored public key than on Tom's, whom he does not trust much at all. There is also a field indicating who can sign other keys within Steve's realm of trust. If Steve receives a key from someone he doesn't know, like Kevin, and the key is signed by Liz, he can look at the field that pertains to whom he trusts to sign other people's keys. If the field indicates that Steve trusts Liz enough to sign another person's key, Steve will accept Kevin's key and communicate with him because Liz is vouching for him. However, if Steve receives a key from Kevin and it is signed by untrustworthy Tom, Steve might choose to not trust Kevin and not communicate with him.

These fields are available for updating and alteration. If one day Steve really gets to know Tom and finds out he is okay after all, he can modify these parameters within PGP and give Tom more trust when it comes to cryptography and secure communication.

Because the web of trust does not have a central leader, such as a CA, certain standardized functionality is harder to accomplish. If Steve were to lose his private key, he would need to notify everyone else trusting his public key that it should no longer be trusted. In a PKI, Steve would only need to notify the CA, and anyone attempting to verify the validity of Steve's public key would be told not to trust it upon looking at the most recently updated CRL. In the PGP world, this is not as centralized and organized. Steve can send out a key revocation certificate, but there is no guarantee it will reach each user's key ring file.

PGP is a public domain software that uses public key cryptography. It has not been endorsed by the NSA, but because it is a great product and free for individuals to use, it has become somewhat of an encryption de facto standard on the Internet.

NOTE PGP is considered a cryptosystem because it has all the necessary components: symmetric key algorithms, asymmetric key algorithms, message digest algorithms, keys, protocols, and the necessary software components.

References

- **MIT Distribution Center for PGP** http://web.mit.edu/network/pgp.html

- *Introduction to Cryptography,* **Chapter 1, "How PGP Works"** www.pgpi.org/doc/pgpintro
- **The International PGP home page** www.pgpi.org

Quantum Cryptography

Gee, cryptography just isn't complex enough. Let's mix some quantum physics in with it.

Today, we have very sophisticated and strong algorithms that are more than strong enough for most uses, even financial transactions and exchanging your secret meatloaf recipe. Some communication data are so critical and so desired by other powerful entities that even our current algorithms may be broken. This type of data might be spy interactions, information warfare, government espionage, and so on. When a whole country wants to break another country's encryption, a great deal of resources will be put behind such efforts—which can put our current algorithms at risk of being broken.

Because of the need to always build a better algorithm, some very smart people have mixed quantum physics and cryptography, which has resulted in a system (if built correctly) that is unbreakable and where any eavesdroppers can be detected. In traditional cryptography, we try to make it very hard for an eavesdropper to break an algorithm and uncover a key, but we cannot detect that an eavesdropper is on the line. In quantum cryptography, however, not only is the encryption very strong, but an eavesdropper *can* be detected.

Quantum cryptography can be carried out using various methods. So, we will walk through one version to give you an idea of how all this works.

Let's say Tom and Kathy are spies and need to send their data back and forth with the assurance it won't be captured. To do so, they need to establish a symmetric encryption key on both ends, one for Tom and one for Kathy.

In ***quantum cryptography,*** photon polarization is commonly used to represent bits (1 or 0). Polarization is the orientation of electromagnetic waves, which is what photons are. Photons are the particles that make up light. The electromagnetic waves have an orientation of horizontal or vertical, or left hand or right hand. Think of a photon as like a jellybean. As a jellybean flies through the air, it can be vertical (standing up straight), horizontal (lying on its back), left handed (tilted to the left), or right handed (tilted to the right). (This is just to conceptually get your head around the idea of polarization.)

Now both Kathy and Tom each have their own photon gun, which they will use to send photons (information) back and forth to each other. They also have a mapping between the polarization of a photon and a binary value. The polarizations can be represented as vertical (|), horizontal (-), left (\), or right (/), and since we only have two values in binary, there must be some overlap.

In this example, a photon with a vertical (|) polarization maps to the binary value of 0. A left polarization (\) maps to 1, a right polarization (/) maps to 0, and a horizontal polarization (-) maps to 1. This mapping (or encoding) is the binary values that make up an encryption key. Tom must have the same mapping to interpret what Kathy sends to him. Tom will use this as his map so when he receives a photon with the polarization of (\), he will write down a 1. When he receives a photon with the polarization of (|), he will write down a 0. He will do this for the whole key and use these values as the key to decrypt a message Kathy sends him.

NOTE If it helps, think about it this way. Tom receives a jellybean that is horizontal and he writes down a 1. The next jellybean he receives is tilted to the right, so he writes down 0. The next jellybean is vertical, so he writes down 1. He does this for all of the jellybeans Kathy sends his way. Now he has a string of binary values that is the encryption key his system will use to decrypt the messages Kathy sends to him.

So they both have to agree upon a key, which is the mapping between the polarization states of the photons and how those states are represented in a binary value. This happens at the beginning of a communication session over a dedicated fiber line. Once the symmetric key is established, it can be used by Kathy and Tom to encrypt and decrypt messages that travel over a more public communication path, like the Internet. The randomness of the polarization and the complexity of creating a symmetric key in this manner help ensure that an eavesdropper will not uncover the encryption key.

Since this type of cryptography is based on quantum physics and not strictly mathematics, the sender and receiver can be confident no eavesdropper is listening to the communication path used to establish their key and that a man-in-the-middle attack is not being carried out. This is because, at the quantum level, even "looking" at an atom or a subatomic particle changes its attributes. This means that if there is an eavesdropper carrying out a passive attack, such as sniffing, the receiver would know because just this simple act changes the characteristics (polarization) of the photons.

NOTE This means that as the jellybeans are sent from Kathy to Tom, if Sean tries to view the jellybeans, the ones that were traveling in a horizontal manner could be tilting left and ones that were traveling in a vertical manner could now be traveling horizontally.

CAUTION This is a very quick explanation of quantum cryptography. To fully understand it, you would need to understand both cryptography and quantum physics in greater depth—and not use jellybeans.

Some people in the industry think quantum cryptography is used between the U.S. White House and the Pentagon and between some military bases and defense contractor locations. This type of information is classified Top Secret by the U.S. government and unless you know the secret handshake and have the right decoder ring, you will not be privy to this type of information.

References

- **Quantum Cryptography** http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci284012,00.html
- **Quantum Cryptography Tutorial** www.cs.dartmouth.edu/~jford/crypto.html
- **Quantum Cryptography** http://en.wikipedia.org/wiki/Quantum_cryptography

Internet Security

Is the Internet tied up into a web?
Response: Well, kind of.

The Web is not the Internet. The Web runs on top of the Internet, in a sense. The Web is the collection of HTTP servers that hold and process web sites we see. The Internet is the collection of physical devices and communication protocols used to transverse these web sites and interact with them. (These issues were touched upon in Chapter 2.) The web sites look the way they do because their creators used a language that dictates the look, feel, and functionality of the page. Web browsers enable users to read web pages by enabling them to request and accept web pages via HTTP, and the user's browser converts the language (HTML, DHTML, and XML) into a format that can be viewed on the monitor. The browser is the user's window to the World Wide Web.

Browsers can understand a variety of protocols and have the capability to process many types of commands, but they do not understand them all. For those protocols or commands the user's browser does not know how to process, the user can download and install a viewer or plug-in, a modular component of code that integrates itself into the system or browser. This is a quick and easy way to expand the functionality of the browser. However, this can cause serious security compromises, because the payload of the module can easily carry viruses and malicious software that users don't discover until it's too late.

Start with the Basics

Why do we connect to the Internet? At first, this seems a basic question, but as we dive deeper into the query, complexity creeps in. We connect to download MP3s, check e-mail, order security books, look at web sites, communicate with friends, and perform various other tasks. But what are we really doing? We are using services provided by a computer's protocols and software. The services may be file transfers provided by FTP, remote connectivity provided by Telnet, Internet connectivity provided by HTTP, secure connections provided by SSL, and much, much more. Without these protocols, there would be no way to even connect to the Internet.

Management needs to decide what functionality employees should have pertaining to Internet use, and the administrator must implement these decisions by controlling services that can be used inside and outside the network. Services can be restricted in various ways, such as: allowing certain services to only run on a particular system and restrict access to that system; employing a secure version of a service; filtering the use of services; or blocking services altogether. These choices determine how secure the site will be and indicate what type of technology is needed to provide this type of protection.

Let's go through many of the technologies and protocols that make up the World Wide Web.

HTTP

TCP/IP is the protocol suite of the Internet, and HTTP is the protocol of the Web. HTTP sits on top of TCP/IP. When a user clicks a link on a web page with her mouse, her browser uses HTTP to send a request to the web server hosting that web site. The web

server finds the corresponding file to that link and sends it to the user via HTTP. So where is TCP/IP in all of this? The TCP protocol controls the handshaking and maintains the connection between the user and the server, and the IP protocol makes sure the file is routed properly throughout the Internet to get from the web server to the user. So, the IP protocol finds the way to get from A to Z, TCP makes sure the origin and destination are correct and that no packets are lost along the way, and, upon arrival at the destination, HTTP presents the payload, which is a web page.

HTTP is a stateless protocol, which means the client and web server make and break a connection for each operation. When a user requests to view a web page, that web server finds the requested web page, presents it to the user, and then terminates the connection. If the user requests a link within the newly received web page, a new connection must be set up, the request goes to the web server, and the web server sends the requested item and breaks the connection. The web server never "remembers" the users that ask for different web pages, because it would have to commit a lot of resources to the effort.

HTTP Secure

HTTP Secure (HTTPS) is HTTP running over SSL. (HTTP works at the application layer and SSL works at the transport layer.) ***Secure Sockets Layer (SSL)*** uses public key encryption and provides data encryption, server authentication, message integrity, and optional client authentication. When a client accesses a web site, that web site may have both secured and public portions. The secured portion would require the user to be authenticated in some fashion. When the client goes from a public page on the web site to a secured page, the web server will start the necessary tasks to invoke SSL and protect this type of communication.

The server sends a message back to the client, indicating a secure session should be established, and the client in response sends its security parameters. The server compares those security parameters to its own until it finds a match. This is the handshaking phase. The server authenticates to the client by sending it a digital certificate, and if the client decides to trust the server, the process continues. The server can require the client to send over a digital certificate for mutual authentication, but that is rare.

The client generates a session key and encrypts it with the server's public key. This encrypted key is sent to the web server, and they both use this symmetric key to encrypt the data they send back and forth. This is how the secure channel is established.

SSL keeps the communication path open until one of the parties requests to end the session. The session is usually ended when the client sends the server a FIN packet, which is an indication to close out the channel.

SSL requires an SSL-enabled server and browser. SSL provides security for the connection but does not offer security for the data once received. This means the data are encrypted while being transmitted, but not after the data are received by a computer. So if a user sends bank account information to a financial institution via a connection protected by SSL, that communication path is protected, but the user must trust the financial institution that receives this information, because at this point, SSL's job is done.

The user can verify that a connection is secure by looking at the URL to see that it includes https://. The user can also check for a padlock or key icon, depending on the browser type, which is shown at the bottom corner of the browser window.

In the protocol stack, SSL lies beneath the application layer and above the network layer. This ensures SSL is not limited to specific application protocols and can still use the communication transport standards of the Internet. Different books and technical resources place SSL at different layers of the OSI model, which may seem confusing at first. But the OSI model is a conceptual construct that attempts to describe the reality of networking. This is like trying to draw nice neat boxes around life—some things don't fit perfectly and hang over the sides. SSL is actually made up of two protocols: one works at the lower end of the session layer, and the other works at the top of the transport layer. This is why one resource will state that SSL works at the session layer and another resource puts it in the transport layer. For the purposes of the CISSP exam, we'll use the latter definition: the SSL protocol works at the transport layer.

Although SSL is almost always used with HTTP, it can also be used with other types of protocols. So if you see a common protocol that is followed by an *s*, that protocol is using SSL to encrypt its data.

Secure HTTP

Though their names are very similar, there is a difference between ***Secure HTTP (S-HTTP)*** and HTTP Secure (HTTPS). S-HTTP is a technology that protects each message sent between two computers, while HTTPS protects the communication channel between two computers, messages and all. HTTPS uses SSL and HTTP to provide a protected circuit between a client and server. So, S-HTTP is used if an individual message needs to be encrypted, but if all information that passes between two computers must be encrypted, then HTTPS is used, which is SSL over HTTP.

References

- ***Security Issues in WWW*** http://nsi.org/Library/Internet/security.htm
- **Keesook J. Han's security links page** www.ece.umn.edu/users/kjhan/security
- **"SSL/TLS Strong Encryption: An Introduction," Apache Software Foundation, with permission from Frederick J. Hirsch** http://httpd.apache.org/docs-2.0/ssl/ssl_intro.html

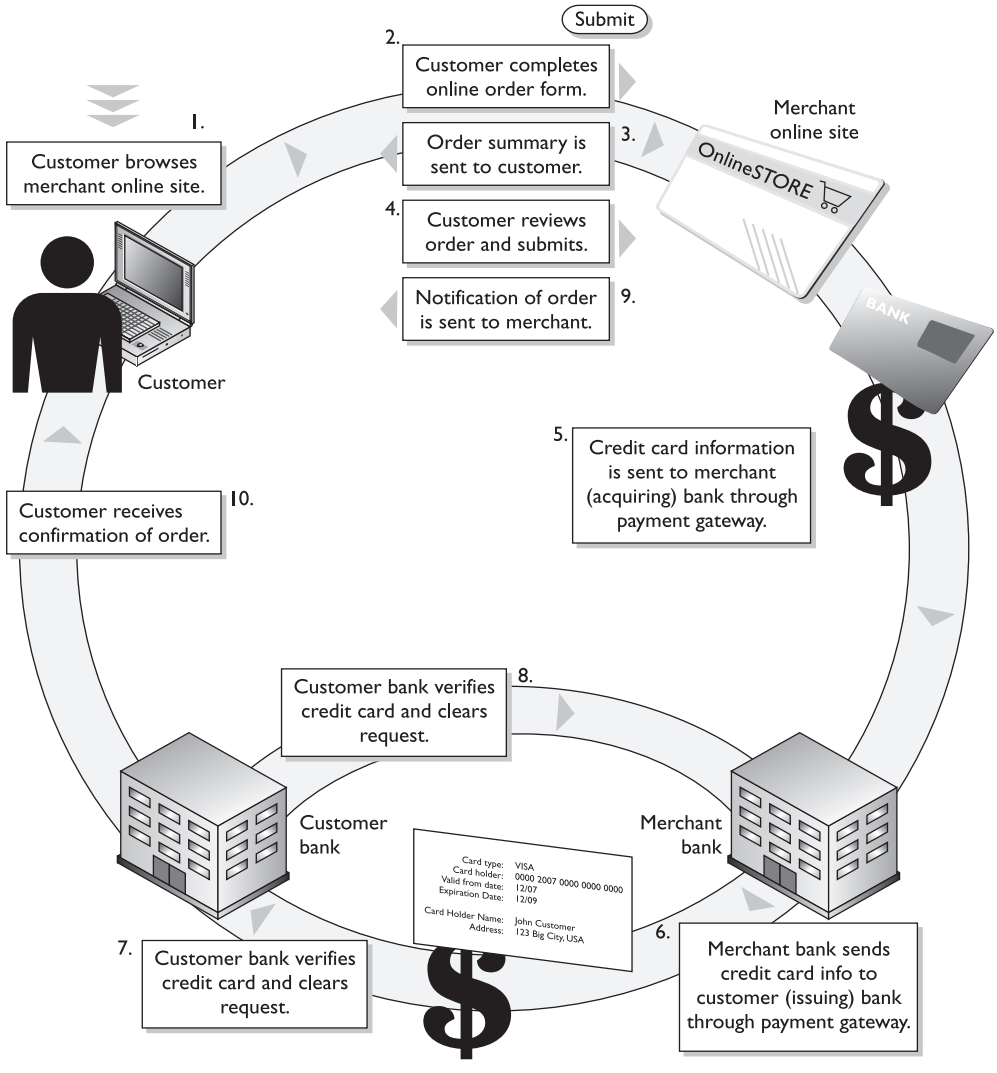
Secure Electronic Transaction

Secure Electronic Transaction (SET) is a security technology proposed by Visa and MasterCard to allow for more secure credit card transaction possibilities than what is currently available. SET has been waiting in the wings for full implementation and acceptance as a standard for quite some time. Although SET provides an effective way of transmitting credit card information, businesses and users do not see it as efficient because it requires more parties to coordinate their efforts, more software installation and configuration for each entity involved, and more effort and cost than the widely used SSL method.

SET is a cryptographic protocol and infrastructure developed to send encrypted credit card numbers over the Internet. The following entities would be involved with a SET transaction, which would require each of them to upgrade their software, and possibly their hardware:

- **Issuer (cardholder's bank)** The financial institution that provides a credit card to the individual.

- **Cardholder** The individual authorized to use a credit card.
- **Merchant** The entity providing goods.
- **Acquirer (merchant's bank)** The financial institution that processes payment cards.
- **Payment gateway** This processes the merchant payment. It may be an acquirer.



To use SET, a user must enter her credit card number into her electronic wallet software. This information is stored on the user's hard drive or on a smart card. The software then creates a public key and a private key that are used specifically for encrypting financial information before it is sent.

Let's say Tanya wants to use her electronic credit card to buy her mother a gift from a web site. When she finds the perfect gift and decides to purchase it, she sends her encrypted credit card information to the merchant's web server. The merchant does not decrypt the credit card information, but instead digitally signs it and sends it on to its processing bank. At the bank, the payment server decrypts the information, verifies that Tanya has the necessary funds, and transfers the funds from Tanya's account to the merchant's account. Then the payment server sends a message to the merchant telling it to finish the transaction, and a receipt is sent to Tanya and the merchant. At each step, an entity verifies a digital signature of the sender and digitally signs the information before it is sent to the next entity involved in the process. This would require all entities to have digital certificates and participate in a PKI.

This is basically a very secure way of doing business over the Internet, but today everyone seems to be happy enough with the security SSL provides. They do not feel motivated enough to move to a different and more encompassing technology. The lack of motivation comes from all of the changes that would need to take place to our current processes and the amount of money these changes would require.

Cookies

Hey, I found a web site that's giving out free cookies!
Response: Great, I'll bring the milk!

Cookies are text files that a browser maintains on a user's hard drive. Cookies have different uses, and some are used for demographic and advertising information. As a user travels from site to site on the Internet, the sites could be writing data to the cookies stored on the user's system. The sites can keep track of the user's browsing and spending habits and the user's specific customization for certain sites. For example, if Emily goes to mainly gardening sites on the Internet, those sites will most likely record this information and the types of items in which she shows most interest. Then, when Emily returns to one of the same or similar sites, it will retrieve her cookies, find she has shown interest in gardening books in the past, and present her with its line of gardening books. This increases the likelihood of Emily purchasing a book of her liking. This is a way of zeroing in on the right marketing tactics for the right person.

The servers at the web site determine how cookies are actually used. When a user adds items to his shopping cart on a site, such data are usually added to a cookie. Then, when the user is ready to check out and pay for his items, all the data in this specific cookie are extracted and the totals are added.

As stated before, HTTP is a stateless protocol, meaning a web server has no memory of any prior connections. This is one reason to use cookies. They retain the memory between HTTP connections by saving prior connection data to the client's computer.

For example, if you carry out your banking activities online, your bank's web server keeps track of your activities through the use of cookies. When you first go to its site and are looking at public information, such as branch locations, hours of operation, and CD rates, no confidential information is being transferred back and forth. Once you make a request to access your bank account, the web server sets up an SSL connection and requires you to send credentials. Once you send your credentials and are authenticated, the server generates a cookie with your authentication and account information in it. The server sends it to your browser, which either saves it to your hard drive or keeps it in memory.

NOTE Some cookies are stored as text files on your hard drive. These files should not contain any sensitive information, such as account numbers and passwords. In most cases, cookies that contain sensitive information stay resident in memory and are not stored on the hard drive.

So, suppose you look at your checking account and do some work there and then request to view your savings account information. The web server sends a request to see if you have been properly authenticated for this activity by checking your cookie.

Most online banking software also periodically requests your cookie, to ensure no man-in-the-middle attacks are going on and that someone else has not hijacked the session.

It is also important to ensure that secure connections time out. This is why cookies have timestamps within them. If you have ever worked on a site that has an SSL connection set up for you and it required you to reauthenticate, the reason is because your session has been idle for a while and, instead of leaving a secure connection open, the web server software closed it out.

A majority of the data within a cookie is meaningless to any entities other than the servers at specific sites, but some cookies can contain usernames and passwords for different accounts on the Internet. The cookies that contain sensitive information should be encrypted by the server at the site that distributes them, but this does not always happen, and a nosey attacker could find this data on the user's hard drive and attempt to use it for mischievous activity. Some people who live on the paranoid side of life do not allow cookies to be downloaded to their systems (controlled through browser security controls). Although this provides a high level of protection against different types of cookie abuse, it also reduces their functionality on the Internet. Some sites require cookies because there is specific data within the cookies that the site must utilize correctly in order to provide the user with the services she requested.

NOTE Some third-party products can limit the type of cookies downloaded, hide the user's identities as he travels from one site to the next, and mask the user's e-mail addresses and the mail servers he uses if he is concerned about concealing his identity and his tracks.

Secure Shell

Secure Shell (SSH) functions as a type of tunneling mechanism that provides terminal-like access to remote computers. SSH is a program and a protocol that can be used to log in to another computer over a network. For example, the program can let Paul, who is on computer A, access computer B's files, run applications on computer B, and retrieve files from computer B without ever physically touching that computer. SSH provides authentication and secure transmission over vulnerable channels like the Internet.

SSH should be used instead of Telnet, FTP, rlogin, rexec, or rsh, which provide the same type of functionality SSH offers but in a much less secure manner. SSH is a program and a set of protocols that work together to provide a secure tunnel between two computers. The two computers go through a handshaking process and exchange (via Diffie-Hellman) a session key that will be used during the session to encrypt and protect the data sent. The steps of an SSH connection are outlined in Figure 8-25.

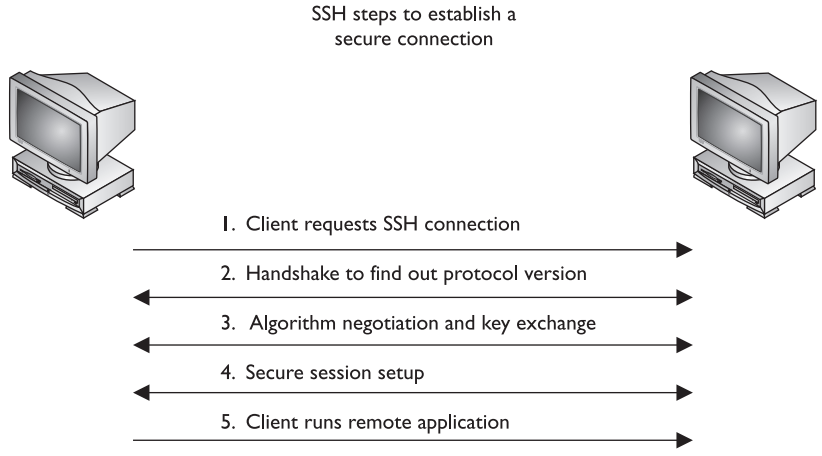
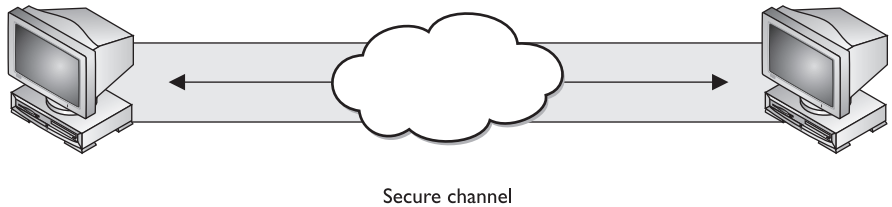


Figure 8-25 SSH is used for remote terminal-like functionality.

Once the handshake takes place and a secure channel is established, the two computers have a pathway to exchange data with the assurance that the information will be encrypted and its integrity will be protected.



References

- **SSH (Secure Shell) FAQ, by Thomas König** www.uni-karlsruhe.de/~ig25/ssh-faq
- **The Secure Shell Frequently Asked Questions, by Anne Carasik and Steve Acheson** www.onsight.com/faq/ssh/ssh-faq.html
- ***Unix System Administration*, Chapter 29, "Secure Shell, SSH," by Frank G. Fiamingo** wks.uts.ohio-state.edu/sysadm_course/html/sysadm-558.html

Internet Security Protocol

Hey, is there a really complex protocol that can provide me with network layer protection? Response: Yep, IPSec.

The ***Internet Protocol Security (IPSec)*** protocol suite provides a method of setting up a secure channel for protected data exchange between two devices. The devices that share this secure channel can be two servers, two routers, a workstation and a server, or two gateways between different networks. IPSec is a widely accepted standard for providing network layer protection. It can be more flexible and less expensive than end-to-end and link encryption methods.

IPSec has strong encryption and authentication methods, and although it can be used to enable tunneled communication between two computers, it is usually employed to establish virtual private networks (VPNs) among networks across the Internet.

IPSec is not a strict protocol that dictates the type of algorithm, keys, and authentication method to use. Rather, it is an open, modular framework that provides a lot of flexibility for companies when they choose to use this type of technology. IPSec uses two basic security protocols: ***Authentication Header (AH)*** and ***Encapsulating Security Payload (ESP).*** AH is the authenticating protocol, and ESP is an authenticating and encrypting protocol that uses cryptographic mechanisms to provide source authentication, confidentiality, and message integrity.

IPSec can work in one of two modes: ***transport mode,*** in which the payload of the message is protected, and ***tunnel mode,*** in which the payload and the routing and header information are protected. ESP in transport mode encrypts the actual message information so it cannot be sniffed and uncovered by an unauthorized entity. Tunnel mode provides a higher level of protection by also protecting the header and trailer data an attacker may find useful. Figure 8-26 shows the high-level view of the steps of setting up an IPSec connection.

Each device will have at least one ***security association (SA)*** for each VPN it uses. The SA, which is critical to the IPSec architecture, is a record of the configurations the device needs to support an IPSec connection. When two devices complete their handshaking process, which means they have agreed upon a long list of parameters they will use to communicate, these data must be recorded and stored somewhere, which is in the SA. The SA can contain the authentication and encryption keys, the agreed-upon algorithms, the key lifetime, and the source IP address. When a device receives a packet via the IPSec protocol, it is the SA that tells the device what to do with the packet. So if device B receives a packet from device C via IPSec, device B will look to the corresponding SA to tell it how to decrypt the packet, how to properly authenticate the source of the packet, which key to use, and how to reply to the message if necessary.

SAs are directional, so a device will have one SA for outbound traffic and a different SA for inbound traffic for each individual communication channel. If a device is connecting to three devices, it will have at least six SAs, one for each inbound and outbound connection per remote device. So how can a device keep all of these SAs organized and ensure that the right SA is invoked for the right connection? With the mighty ***secu-***

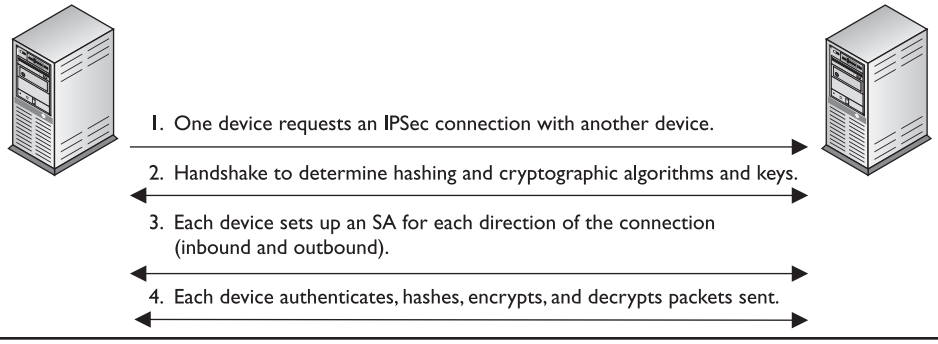


Figure 8-26 Steps that two computers follow when using IPSec

rity parameter index (SPI), that's how. Each device has an SPI that keeps track of the different SAs and tells the device which one is appropriate to invoke for the different packets it receives. The SPI value is in the header of an IPSec packet, and the device reads this value to tell it which SA to consult, as depicted in Figure 8-27.

IPSec can authenticate the sending devices of the packet by using MAC (covered in the earlier section, "The One-Way Hash"). The ESP protocol can provide authentication, integrity, and confidentiality if the devices are configured for this type of functionality. So if a company just needs to make sure it knows the source of the sender and must be assured of the integrity of the packets, it would choose to use AH. If the company would like to use these services and also have confidentiality, it would use the ESP protocol because it provides encryption functionality. In most cases, the reason ESP is employed is because the company must set up a secure VPN connection.

It may seem odd to have two different protocols that provide overlapping functionality. AH provides authentication and integrity, and ESP can provide those two functions *and* confidentiality. Why even bother with AH then? In most cases, the reason has to do with whether the environment is using network address translation (NAT). IPSec will generate an integrity check value (ICV), which is really the same thing as a MAC value, over a portion of the packet. Remember that the sender and receiver generate their own values. In IPSec, it is called an ICV value. The receiver compares her ICV value with the one sent by the sender. If the values match, the receiver can be assured the packet has not been modified during transmission. If the values are different, the packet has been altered and the receiver discards the packet.

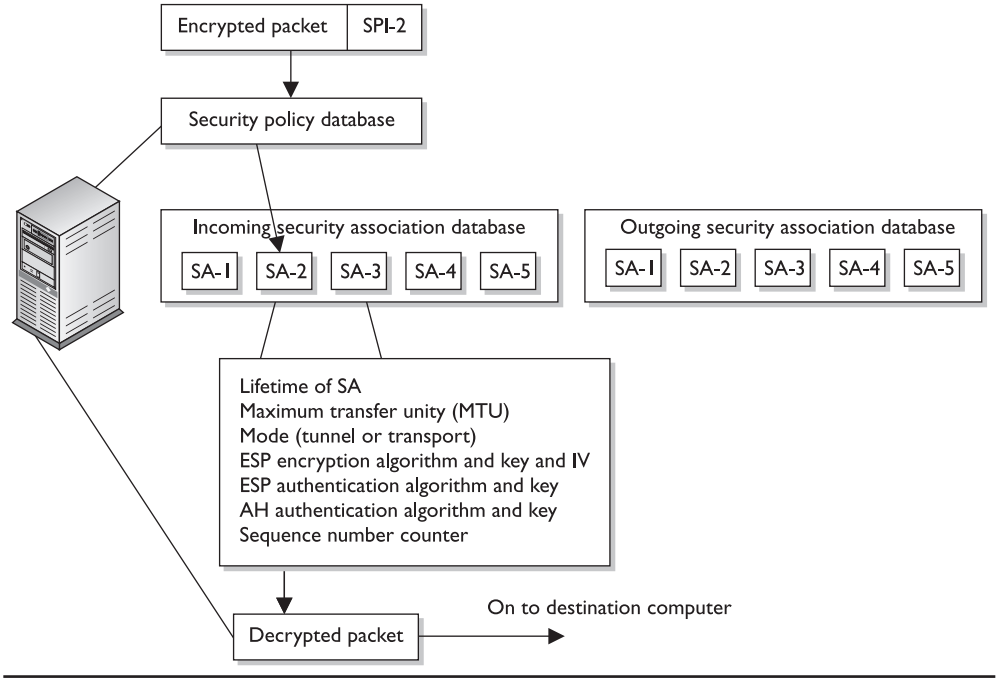


Figure 8-27 The SPI and SA help the system process IPSec packets.

The AH protocol calculates this ICV over the data payload, transport, and network headers. If the packet then goes through a NAT device, the NAT device changes the IP address of the packet. That is its job. This means a portion of the data (network header) that was included to calculate the ICV value has now changed, and the receiver will generate an ICV value that is different from the one sent with the packet, which means the packet will be discarded automatically.

The ESP protocol follows similar steps, except it does not include the network header portion when calculating its ICV value. When the NAT device changes the IP address, it will not affect the receiver's ICV value because it does not include the network header when calculating the ICV. The differences are shown in Figure 8-28.

Because IPSec is a framework, it does not dictate which hashing and encryption algorithms are to be used or how keys are to be exchanged between devices. Key management can be handled manually or automated by a key management protocol. The de facto standard for IPSec is to use Internet Key Exchange (IKE), which is a combination of the ISAKMP and OAKLEY protocols. The ***Internet Security Association and Key Management Protocol (ISAKMP)*** is a key exchange architecture that is independent of the type of keying mechanisms used. Basically, ISAKMP provides the framework of what can be negotiated to set up an IPSec connection (algorithms, protocols, modes, keys). The OAKLEY protocol is the one that carries out the negotiation process. You can think of ISAKMP as providing the playing field (the infrastructure) and OAKLEY as the guy running up and down the playing field (carrying out the steps of the negotiation).

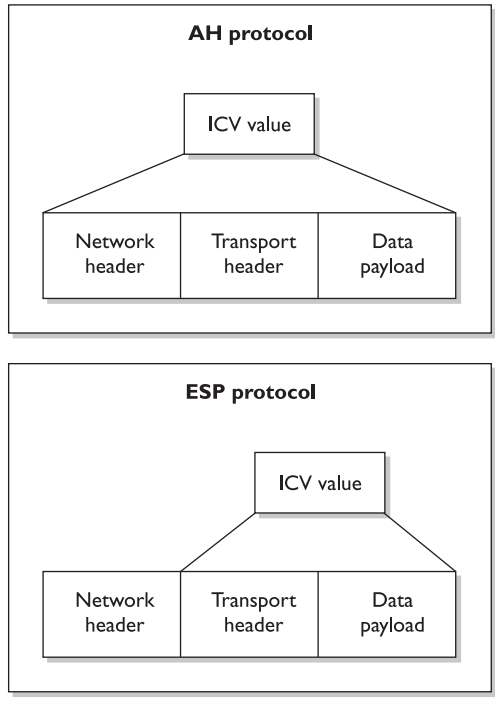


Figure 8-28 AH and ESP use different portions of the packet to calculate the ICVs.

NOTE Simple Key Management Protocol for IP (SKIP) is another key exchange protocol that provides basically the same functionality as IKE. It is important to know that all of these protocols work at the network layer.

IPSec is very complex with all of its components and possible configurations. This complexity is what provides for a great degree of flexibility, because a company has many different configuration choices to achieve just the right level of protection. If this is all new to you and still confusing, please review one or more of the following references to help fill in the gray areas.

References

- **"A Cryptographic Evaluation of IPSec," by N. Ferguson and Bruce Schneier** www.schneier.com/paper-ipsec.html
- **"An Introduction to IP Security (IPSec) Encryption," Cisco Systems, Inc.** www.cisco.com/warp/public/105/IPSECpart1.html

Attacks

Eavesdropping and sniffing data as it passes over a network are considered ***passive attacks*** because the attacker is not affecting the protocol, algorithm, key, message, or any parts of the encryption system. Passive attacks are hard to detect, so in most cases methods are put in place to try to prevent them rather than detect and stop them.

Altering messages, modifying system files, and masquerading as another individual are acts that are considered ***active attacks*** because the attacker is actually doing something instead of sitting back and gathering data. Passive attacks are usually used to gain information prior to carrying out an active attack. The following sections address some active attacks that relate to cryptography.

Cipher-Only Attack

In this type of attack, the attacker has the ciphertext of several messages. Each of the messages has been encrypted using the same encryption algorithm. The attacker's goal is to discover the key used in the encryption process. Once the attacker figures out the key, she can decrypt all other messages encrypted with the same key.

A ciphertext-only attack is the most common type of active attack because it is very easy to get ciphertext by sniffing someone's traffic, but it is the hardest attack to actually be successful at because the attacker has so little information about the encryption process.

Known-Plaintext Attacks

In known-plaintext attacks, the attacker has the plaintext and ciphertext of one or more messages. Again, the goal is to discover the key used to encrypt the messages so other messages can be deciphered and read.

Messages usually start with the same type of beginning and close with the same type of ending. An attacker might know that each message a general sends out to his

commanders always starts with certain greetings and ends with specific salutations and the general's name and contact information. In this instance, the attacker has some of the plaintext (the data that are the same on each message) and can capture an encrypted message, and therefore capture the ciphertext. Once a few pieces of the puzzle are discovered, the rest is accomplished by reverse-engineering, frequency analysis, and brute force attempts. Known-plaintext attacks were used by the United States against the Germans and the Japanese during World War II.

Chosen-Plaintext Attacks

In chosen-plaintext attacks, the attacker has the plaintext and ciphertext, but can choose the plaintext that gets encrypted to see the corresponding ciphertext. This gives her more power and possibly a deeper understanding of the way the encryption process works so she can gather more information about the key being used. Once the key is discovered, other messages encrypted with that key can be decrypted.

How would this be carried out? I can e-mail a message to you that I think you not only will believe, but that you will also panic about, encrypt, and send to someone else. Suppose I send you an e-mail that states, "The meaning of life is 42." You may think you have received an important piece of information that should be concealed from others, everyone except your friend Bob, of course. So you encrypt my message and send it to Bob. Meanwhile I am sniffing your traffic and now have a copy of the plaintext of the message, because I wrote it, and a copy of the ciphertext.

Chosen-Ciphertext Attacks

In chosen-ciphertext attacks, the attacker can choose the ciphertext to be decrypted and has access to the resulting decrypted plaintext. Again, the goal is to figure out the key. This is a harder attack to carry out compared to the previously mentioned attacks, and the attacker may need to have control of the system that contains the cryptosystem.

Public vs. Secret Algorithms

The public mainly uses algorithms that are known and understood versus the secret algorithms where the internal processes and functions are not released to the public. In general, cryptographers in the public sector feel as though the strongest and best-engineered algorithms are the ones released for peer review and public scrutiny, because a thousand brains are better than five, and many times some smarty-pants within the public population can find problems within an algorithm that the developers did not think of. This is why vendors and companies have competitions to see if anyone can break their code and encryption processes. If someone does break it, that means the developers must go back to the drawing board and strengthen this or that piece.

Not all algorithms are released to the public, such as the ones developed by the NSA. Because the sensitivity level of what the NSA encrypts is so important, it wants as much of the process to be as secret as possible. The fact that the NSA does not release its algorithms for public examination and analysis does not mean its algorithms are weak. Its algorithms are developed, reviewed, and tested by many of the top cryptographic smarty-pants around, and are of very high quality.

NOTE All of these attacks have a derivative form, the names of which are the same except for putting the word "adaptive" in front of them: such as adaptive chosen-plaintext and adaptive chosen-ciphertext. What this means is that the attacker can carry out one of these attacks and, depending upon what she gleaned from that first attack, modify her next attack. This is the process of reverse-engineering or cryptanalysis attacks: using what you learned to improve your next attack.

Differential Cryptanalysis

This type of attack also has the goal of uncovering the key that was used for encryption purposes. It was invented in 1990 as an attack against DES, and it turned out to be an effective and successful attack against DES and other block algorithms.

The attacker takes two messages of plaintext and follows the changes that take place to the blocks as they go through the different S-boxes. (Each message is being encrypted with the same key.) The differences identified in the resulting ciphertext values are used to map probability values to different possible key values. The attacker continues this process with several more sets of messages and reviews the common key probability values. One key will continue to show itself as the most probable key used in the encryption processes. Since the attacker chooses the different plaintext messages for this attack, it is considered to be a type of chosen-plaintext attack.

Linear Cryptanalysis

Linear cryptanalysis is another type of attack that carries out functions to identify the highest probability of a specific key employed during the encryption process using a block algorithm. The attacker carries out a known-plaintext attack on several different messages encrypted with the same key. The more messages the attacker can use and put through this type of attack, the higher the confidence level in the probability of a specific key value.

The attacker evaluates the input and output values for each S-box. He evaluates the probability of input values ending up in a specific combination. Identifying specific output combinations allows him to assign probability values to different keys until one shows a continual pattern of having the highest probability.

Side-Channel Attacks

All of the attacks we have covered thus far have been based mainly on the mathematics of cryptography. Using plaintext and ciphertext involves high-powered mathematical tools that are needed to uncover the key used in the encryption process.

But what if we took a different approach? Let's say we see something that looks like a duck, walks like a duck, sounds like a duck, swims in water, and eats bugs and small fish. We could confidently conclude that this is a duck. Similarly, in cryptography, we can review facts and infer the value of an encryption key. For example, we could detect how much power consumption is used for encryption and decryption (the fluctuation of electronic voltage). We could also intercept the radiation emissions released and then calculate how long the processes take. Looking around the cryptosystem, or its attributes and characteristics, is different from looking into the cryptosystem and trying to defeat it through mathematical computations.

If I want to figure out what you do for a living, but I don't want you to know I am doing this type of reconnaissance work, I won't ask you directly. Instead, I will find out when you go to work and come home, the types of clothing you wear, the items you carry, whom you talk to… or I can just follow you to work. These are examples of side channels.

So, in cryptography, gathering "outside" information with the goal of uncovering the encryption key is just another way of attacking a cryptosystem.

An attacker could measure power consumption, radiation emissions, and the time it takes for certain types of data processing. With this information, he can work backward by reverse-engineering the process to uncover an encryption key or sensitive data. A power attack reviews the amount of heat released. This type of attack has been successful in uncovering confidential information from smart cards. In 1995, RSA private keys were uncovered by measuring the relative time cryptographic operations took.

The idea is that, instead of attacking a device head on, just watch how it performs to figure out how it works. In biology, scientists can choose to carry out a noninvasive experiment, which will watch an organism eat, sleep, mate, and so on. This type of approach learns about the organism through understanding its behaviors instead of killing it and looking at it from the inside out.

Replay Attacks

A big concern in distributed environments is the ***replay attack,*** in which an attacker captures some type of data and resubmits it with the hopes of fooling the receiving device into thinking it is legitimate information. Many times, the data captured and resubmitted are authentication information, and the attacker is trying to authenticate herself as someone else to gain unauthorized access.

Timestamps and sequence numbers are two countermeasures to replay attacks. Packets can contain sequence numbers, so each machine will expect a specific number on each receiving packet. If a packet has a sequence number that has been previously used, this is an indication of a replay attack. Packets can also be timestamped. A threshold can be set on each computer to only accept packets within a certain timeframe. If a packet is received that is past this threshold, it can help identify a replay attack.

Just in case there aren't enough attacks here for you, we have three more, which are quickly introduced in the following sections.

Algebraic Attacks

Algebraic attacks analyze the vulnerabilities in the mathematics used within the algorithm and exploit the intrinsic algebraic structure. For instance, attacks on the "textbook" version of the RSA cryptosystem exploit properties of the algorithm such as the fact that the encryption of a raw "0" message is "0".

Analytic

Analytic attacks identify algorithm structural weaknesses or flaws, as opposed to brute force attacks which simply exhaust all possibilities without respect to the specific properties of the algorithm. Examples = Double DES attack and RSA factoring attack.

Statistical

Statistical attacks identify statistical weaknesses in algorithm design for exploitation—for example, if statistical patterns are identified, as in the number of 0s compared to the number of 1s. For instance, a random number generator may be biased. If keys are taken directly from the output of the RNG, then the distribution of keys would also be biased. The statistical knowledge about the bias could be used to reduce the search time for the keys.

References

- ***Wikipedia* entry for ciphertext-only attack** www.answers.com/topic/ciphertext-only-attack
- **Frequently Asked Questions about Today's Cryptography, Version 4.1, Section 2.4.2, "What Are Some of the Basic Types of Cryptanalytic Attacks?" by RSA Laboratories** www.rsasecurity.com/rsalabs/node.asp?id=2201
- **"Linear Cryptanalysis: A Literature Survey," by Terry Ritter** www.ciphersbyritter.com/RES/LINANA.HTM
- **"Linear Cryptanalysis of Block Ciphers," by Edward Schaefer** http://math.scu.edu/~eschaefe/linear.pdf
- **"Introduction to Side Channel Attacks," by Hagai Bar-El, Discretix Technologies Ltd.** www.discretix.com/PDF/Introduction%20to%20Side%20Channel%20Attacks.pdf

Summary

Cryptography has been used in one form or another for over 4000 years, and the attacks on cryptography have probably been in place for 3999 years and 364 days. As one group of people works to find new ways to hide and transmit secrets, another group of people is right on their heels finding holes in the newly developed ideas and products. This can be viewed as evil and destructive behavior, or as the thorn in the side of the computing world that pushes it to build better and more secure products and environments.

Cryptographic algorithms provide the underlying tools to most security protocols used in today's infrastructures. The algorithms work off of mathematical functions and provide various types of functionality and levels of security. A big leap was made when encryption went from purely symmetric key use to public key cryptography. This evolution provided users and maintainers much more freedom and flexibility when it came to communicating with a variety of users all over the world.

Encryption can be supplied at different layers of the OSI model by a range of applications, protocols, and mechanisms. Today, not much thought has to be given to cryptography and encryption because it is taken care of in the background by many operating systems, applications, and protocols. However, for administrators who maintain these environments, for security professionals who propose and implement security solutions, and for those interested in obtaining a CISSP certification, knowing the ins and outs of cryptography is essential.

Quick Tips

- Cryptography is the science of protecting information by encoding it into an unreadable format.
- The most famous rotor encryption machine is the Enigma used by the Germans in WWII.
- A readable message is in a form called plaintext, and once it is encrypted, it is in a form called ciphertext.
- Cryptographic algorithms are the mathematical rules that dictate the functions of enciphering and deciphering.
- Cryptanalysis is the study of breaking cryptosystems.
- Nonrepudiation is a service that ensures the sender cannot later falsely deny sending a message.
- Key clustering is an instance in which two different keys generate the same ciphertext from the same plaintext.
- The range of possible keys is referred to as the keyspace. A larger keyspace and the full use of the keyspace allow for more random keys to be created. This provides more protection.
- The two basic types of encryption mechanisms used in symmetric ciphers are substitution and transposition. Substitution ciphers change a character (or bit) out for another, while transposition ciphers scramble the characters (or bits).
- A polyalphabetic cipher uses more than one alphabet to defeat frequency analysis.
- Steganography is a method of hiding data within another media type, such as a graphic, WAV file, or document. This method is used to hide the existence of the data.
- A key is a random string of bits inserted into an encryption algorithm. The result determines what encryption functions will be carried out on a message and in what order.
- In symmetric key algorithms, the sender and receiver use the same key for encryption and decryption purposes.
- In asymmetric key algorithms, the sender and receiver use different keys for encryption and decryption purposes.
- Symmetric key processes provide barriers of secure key distribution and scalability. However, symmetric key algorithms perform much faster than asymmetric key algorithms.
- Symmetric key algorithms can provide confidentiality, but not authentication or nonrepudiation.
- Examples of symmetric key algorithms include DES, 3DES, Blowfish, IDEA, RC4, RC5, RC6, and AES.

- Asymmetric algorithms are used to encrypt keys, and symmetric algorithms are used to encrypt bulk data.
- If a user encrypts data with his private key, that data can only be decrypted by his public key.
- Asymmetric key algorithms are much slower than symmetric key algorithms, but can provide authentication and nonrepudiation services.
- Examples of asymmetric key algorithms include RSA, ECC, Diffie-Hellman, El Gamal, Knapsack, and DSA.
- Two main types of symmetric algorithms are stream and block ciphers. Stream ciphers use a keystream generator and encrypt a message one bit at a time. A block cipher divides the message into groups of bits and encrypts them.
- Block ciphers are usually implemented in software, and stream ciphers are usually implemented in hardware.
- Many algorithms are publicly known, so the secret part of the process is the key. The key provides the necessary randomization to encryption.
- Data Encryption Standard (DES) is a block cipher that divides a message into 64-bit blocks and employs S-box-type functions on them.
- Because technology has allowed the DES keyspace to be successfully broken, Triple-DES (3DES) was developed to be used instead. 3DES uses 48 rounds of computation and up to three different keys.
- International Data Encryption Algorithm (IDEA) is a symmetric block cipher with a key of 128 bits.
- RSA is an asymmetric algorithm developed by Rivest, Shamir, and Adleman and is the de facto standard for digital signatures.
- Elliptic curve cryptosystems (ECCs) are used as asymmetric algorithms and can provide digital signature, secure key distribution, and encryption functionality. They use much less resources, which makes them better for wireless device and cell phone encryption use.
- When symmetric and asymmetric key algorithms are used together, this is called a hybrid system. The asymmetric algorithm encrypts the symmetric key, and the symmetric key encrypts the data.
- A session key is a symmetric key used by the sender and receiver of messages for encryption and decryption purposes. The session key is only good while that communication session is active and then it is destroyed.
- A public key infrastructure (PKI) is a framework of programs, procedures, communication protocols, and public key cryptography that enables a diverse group of individuals to communicate securely.
- A certificate authority (CA) is a trusted third party that generates and maintains user certificates, which hold their public keys.
- The CA uses a certification revocation list (CRL) to keep track of revoked certificates.

- A certificate is the mechanism the CA uses to associate a public key to a person's identity.
- A registration authority (RA) validates the user's identity and then sends the request for a certificate to the CA. The RA cannot generate certificates.
- A one-way function is a mathematical function that is easier to compute in one direction than in the opposite direction.
- RSA is based on a one-way function that factors large numbers into prime numbers. Only the private key knows how to use the trapdoor and decrypt messages that were encrypted with the corresponding public key.
- Hashing algorithms provide data integrity only.
- When a hash algorithm is applied to a message, it produces a message digest, and this value is signed with a private key to produce a digital signature.
- Some examples of hashing algorithms include SHA-1, MD2, MD4, MD5, and HAVAL.
- HAVAL produces a variable-length hash value, whereas the others produce a fixed-length value.
- SHA-1 produces a 160-bit hash value and is used in DSS.
- A birthday attack is an attack on hashing functions through brute force. The attacker tries to create two messages with the same hashing value.
- A one-time pad uses a pad with random values that are XORed against the message to produce ciphertext. The pad is at least as long as the message itself and is used once and then discarded.
- A digital signature is the result of a user signing a hash value with a private key. It provides authentication, data integrity, and nonrepudiation. The act of signing is the actual encryption of the value with the private key.
- Examples of algorithms used for digital signatures include RSA, El Gamal, ECDSA, and DSA.
- Key management is one of the most challenging pieces of cryptography. It pertains to creating, maintaining, distributing, and destroying cryptographic keys.
- The Diffie-Hellman protocol is a key agreement protocol and does not provide encryption for data and cannot be used in digital signatures.
- Link encryption encrypts the entire packet, including headers and trailers, and has to be decrypted at each hop. End-to-end encryption does not encrypt the headers and trailers, and therefore does not need to be decrypted at each hop.
- Privacy-Enhanced Mail (PEM) is an Internet standard that provides secure e-mail over the Internet by using encryption, digital signatures, and key management.
- Message Security Protocol (MSP) is the military's PEM.
- Pretty Good Privacy (PGP) is an e-mail security program that uses public key encryption. It employs a web of trust instead of the hierarchical structure used in PKI.

- S-HTTP provides protection for each message sent between two computers, but not the actual link. HTTPS protects the communication channel. HTTPS is HTTP that uses SSL for security purposes.
- Secure Electronic Transaction (SET) is a proposed electronic commerce technology that provides a safer method for customers and merchants to perform transactions over the Internet.
- In IPSec, AH provides integrity and authentication, and ESP provides those plus confidentiality.
- IPSec protocols can work in transport mode (the data payload is protected) or tunnel mode (the payload and headers are protected).
- IPSec uses IKE as its key exchange protocol. IKE is the de facto standard and is a combination of ISAKMP and OAKLEY.
- DEA is the algorithm used for the DES standard.

Questions

Please remember that these questions are formatted and asked in a certain way for a reason. Keep in mind that the CISSP exam is asking questions at a conceptual level. Questions may not always have the perfect answer, and the candidate is advised against always looking for the perfect answer. The candidate should look for the best answer in the list.

1. What is the goal of cryptanalysis?
   - **A.** To determine the strength of an algorithm
   - **B.** To increase the substitution functions in a cryptographic algorithm
   - **C.** To decrease the transposition functions in a cryptographic algorithm
   - **D.** To determine the permutations used
2. The frequency of brute force attacks has increased because:
   - **A.** The use of permutations and transpositions in algorithms has increased.
   - **B.** As algorithms get stronger, they get less complex, and thus more susceptible to attacks.
   - **C.** Processor speed and power has increased.
   - **D.** Key length reduces over time.
3. Which of the following is not a property or characteristic of a one-way hash function?
   - **A.** It converts a message of arbitrary length into a value of fixed length.
   - **B.** Given the digest value, it should be computationally infeasible to find the corresponding message.
   - **C.** It should be impossible or rare to derive the same digest from two different messages.
   - **D.** It converts a message of fixed length to an arbitrary length value.

4. What would indicate that a message had been modified?
   - **A.** The public key has been altered.
   - **B.** The private key has been altered.
   - **C.** The message digest has been altered.
   - **D.** The message has been encrypted properly.
5. Which of the following is a U.S. federal government algorithm developed for creating secure message digests?
   - **A.** Data Encryption Algorithm
   - **B.** Digital Signature Standard
   - **C.** Secure Hash Algorithm
   - **D.** Data Signature Algorithm
6. Which of the following best describes the difference between HMAC and CBC-MAC?
   - **A.** HMAC creates a message digest and is used for integrity; CBC-MAC is used to encrypt blocks of data for confidentiality.
   - **B.** HMAC uses a symmetric key and a hashing algorithm; CBC-MAC uses the first block for the checksum.
   - **C.** HMAC provides integrity and data origin authentication; CBC-MAC uses a block cipher for the process of creating a MAC.
   - **D.** HMAC encrypts a message with a symmetric key and then puts the result through a hashing algorithm; CBC-MAC encrypts the whole message.
7. What is an advantage of RSA over the DSA?
   - **A.** It can provide digital signature and encryption functionality.
   - **B.** It uses fewer resources and encrypts faster because it uses symmetric keys.
   - **C.** It is a block cipher rather than a stream cipher.
   - **D.** It employs a one-time encryption pad.
8. Many countries restrict the use or exportation of cryptographic systems. What is the reason given when these types of restrictions are put into place?
   - **A.** Without standards, there would be many interoperability issues when trying to employ different algorithms in different programs.
   - **B.** The systems can be used by some countries against their local people.
   - **C.** Criminals could use encryption to avoid detection and prosecution.
   - **D.** Laws are way behind, so adding different types of encryption would confuse the laws more.
9. What is used to create a digital signature?
   - **A.** The receiver's private key
   - **B.** The sender's public key

C. The sender's private key

D. The receiver's public key

10. Which of the following best describes a digital signature?

A. A method of transferring a handwritten signature to an electronic document

B. A method to encrypt confidential information

C. A method to provide an electronic signature and encryption

D. A method to let the receiver of the message prove the source and integrity of a message

11. How many bits make up the effective length of the DES key?

A. 56

B. 64

C. 32

D. 16

12. Why would a certificate authority revoke a certificate?

A. If the user's public key has become compromised

B. If the user changed over to using the PEM model that uses a web of trust

C. If the user's private key has become compromised

D. If the user moved to a new location

13. What does DES stand for?

A. Data Encryption System

B. Data Encryption Standard

C. Data Encoding Standard

D. Data Encryption Signature

14. Which of the following best describes a certificate authority?

A. An organization that issues private keys and the corresponding algorithms

B. An organization that validates encryption processes

C. An organization that verifies encryption keys

D. An organization that issues certificates

15. What does DEA stand for?

A. Data Encoding Algorithm

B. Data Encoding Application

C. Data Encryption Algorithm

D. Digital Encryption Algorithm

16. Who was involved in developing the first public key algorithm?
    A. Adi Shamir
    B. Ross Anderson
    C. Bruce Schneier
    D. Martin Hellman
17. What process usually takes place after creating a DES session key?
    A. Key signing
    B. Key escrow
    C. Key clustering
    D. Key exchange
18. DES performs how many rounds of permutation and substitution?
    A. 16
    B. 32
    C. 64
    D. 56
19. Which of the following is a true statement pertaining to data encryption when it is used to protect data?
    A. It verifies the integrity and accuracy of the data.
    B. It requires careful key management.
    C. It does not require much system overhead in resources.
    D. It requires keys to be escrowed.
20. If different keys generate the same ciphertext for the same message, what is this called?
    A. Collision
    B. Secure hashing
    C. MAC
    D. Key clustering
21. What is the definition of an algorithm's work factor?
    A. The time it takes to encrypt and decrypt the same plaintext
    B. The time it takes to break the encryption
    C. The time it takes to implement 16 rounds of computation
    D. The time it takes to apply substitution functions
22. What is the primary purpose of using one-way hashing on user passwords?
    A. It minimizes the amount of primary and secondary storage needed to store passwords.
    B. It prevents anyone from reading passwords in plaintext.

C. It avoids excessive processing required by an asymmetric algorithm.

D. It prevents replay attacks.

23. Which of the following is based on the fact that it is hard to factor large numbers into two original prime numbers?

A. ECC

B. RSA

C. DES

D. Diffie-Hellman

24. Which of the following describes the difference between the Data Encryption Standard and the Rivest-Shamir-Adleman algorithm?

A. DES is symmetric, while RSA is asymmetric.

B. DES is asymmetric, while RSA is symmetric.

C. They are hashing algorithms, but RSA produces a 160-bit hashing value.

D. DES creates public and private keys, while RSA encrypts messages.

25. Which of the following uses a symmetric key and a hashing algorithm?

A. HMAC

B. Triple-DES

C. ISAKMP-OAKLEY

D. RSA

Answers

1. **A.** Cryptanalysis is the process of trying to reverse-engineer a cryptosystem with the possible goal of uncovering the key used. Once this key is uncovered, all other messages encrypted with this key can be accessed. Cryptanalysis is carried out by the white hats to test the strength of the algorithm.
2. **C.** A brute force attack is resource-intensive. It guesses values until the correct one is obtained. As computers have more powerful processors added to them, attackers can carry out more powerful brute force attacks.
3. **D.** A hashing algorithm will take a string of variable length, the message can be of any size, and compute a fixed-length value. The fixed-length value is the message digest. The MD family creates the fixed-length value of 128 bits, and SHA creates one of 160 bits.
4. **C.** Hashing algorithms generate message digests to detect whether modification has taken place. The sender and receiver independently generate their own digests, and the receiver compares these values. If they differ, the receiver knows the message has been altered.
5. **C.** SHA was created to generate secure message digests. Digital Signature Standard (DSS) is the standard to create digital signatures, which dictates

that SHA must be used. DSS also outlines the digital signature algorithms that can be used with SHA: RSA, DSA, ECDSA.

6. **C.** In an HMAC operation, a message is concatenated with a symmetric key and the result is put through a hashing algorithm. This provides integrity and system or data authentication. CBC-MAC uses a block cipher to create a MAC, which is the last block of ciphertext.
7. **A.** RSA can be used for data encryption, key exchange, and digital signatures. DSA can be used only for digital signatures.
8. **C.** The U.S. government has greatly reduced its restrictions on cryptography exportation, but there are still some restrictions in place. Products that use encryption cannot be sold to any country the United States has declared is supporting terrorism. The fear is that the enemies of the country would use encryption to hide their communication, and the government would be unable to break this encryption and spy on their data transfers.
9. **C.** A digital signature is a message digest that has been encrypted with the sender's private key. A sender, or anyone else, should never have access to the receiver's private key.
10. **D.** A digital signature provides authentication (knowing who really sent the message), integrity (because a hashing algorithm is involved), and nonrepudiation (the sender cannot deny sending the message).
11. **A.** DES has a key size of 64 bits, but 8 bits are used for parity, so the true key size is 56 bits. Remember that DEA is the algorithm used for the DES standard, so DEA also has a true key size of 56 bits, because we are actually talking about the same algorithm here. DES is really the standard and DEA is the algorithm. We just call it DES in the industry because it is easier.
12. **C.** The reason a certificate is revoked is to warn others who use that person's public key that they should no longer trust the public key because, for some reason, that public key is no longer bound to that particular individual's identity. This could be because an employee left the company, or changed his name and needed a new certificate, but most likely it is because the person's private key was compromised.
13. **B.** Data Encryption Standard was developed by NIST and the NSA to be used to encrypt sensitive but unclassified government data.
14. **D.** A registration authority (RA) accepts a person's request for a certificate and verifies that person's identity. Then the RA sends this request to a certificate authority (CA), which generates and maintains the certificate. Some companies are in business solely for this purpose—Entrust and VeriSign are just two examples.
15. **C.** DEA is the algorithm that fulfilled the DES standard. So DEA has all of the attributes of DES: a symmetric block cipher that uses 64-bit blocks, 16 rounds, and a 56-bit key.

16. **D.** The first released public key cryptography algorithm was developed by Whitfield Diffie and Martin Hellman.
17. **D.** After a session key has been created, it must be exchanged securely. In most cryptosystems, an asymmetric key (the receiver's public key) is used to encrypt this session key, and it is sent to the receiver.
18. **A.** DES carries out 16 rounds of mathematical computation on each 64-bit block of data it is responsible for encrypting. A round is a set of mathematical formulas used for encryption and decryption processes.
19. **B.** Data encryption always requires careful key management. Most algorithms are so strong today it is much easier to go after key management rather than launch a brute force attack. Hashing algorithms are used for data integrity, encryption does require a good amount of resources, and keys do not have to be escrowed for encryption.
20. **D.** Message A was encrypted with key A and the result is ciphertext Y. If that same message A were encrypted with key B, the result should not be ciphertext Y. The ciphertext should be different since a different key was used. But if the ciphertext is the same, this occurrence is referred to as key clustering.
21. **B.** The work factor of a cryptosystem is the amount of time and resources necessary to break the cryptosystem or its encryption process. The goal is to make the work factor so high that an attacker could not be successful at this type of attack.
22. **B.** Passwords are usually run through a one-way hashing algorithm so the actual password is not transmitted across the network or stored on the authentication server in plaintext. This greatly reduces the risk of an attacker being able to obtain the actual password.
23. **B.** The RSA algorithm's security is based on the difficulty of factoring large numbers into their original prime numbers. This is a one-way function. It is easier to calculate the product than it is to identify the prime numbers used to generate that product.
24. **A.** DES is a symmetric algorithm. RSA is a public key algorithm. DES is used to encrypt data, and RSA is used to create public/private key pairs.
25. **A.** When an HMAC function is used, a symmetric key is combined with the message, and then that result is put though a hashing algorithm. The result is an HMAC value. HMAC provides data origin authentication and data integrity.