

Cryptography

In this chapter, you will learn about the following items:

- History of cryptography
- Cryptography components and their relationships
- Government involvement in cryptography
- Symmetric and asymmetric key cryptosystems
- 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 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 network communication paths.

Although the ultimate goal of cryptography, and the mechanisms that make it up, is to hide information from unauthorized individuals, most algorithms can be broken and the information can be revealed if the attacker has enough time, desire, and resources. So a more realistic goal of cryptography is to make obtaining the information too work-intensive to be worth it to the attacker.

The first encryption methods date back to 4,000 years ago and were considered more of an ancient art. As encryption evolved, it was mainly used to pass messages through hostile environments of war, crisis, and for negotiation processes between conflicting groups of people. Throughout history, individuals and governments have worked to protect communication by encrypting it. As time went on, the encryption algorithms and the devices that used them increased in complexity, new methods and algorithms were continually introduced, and it became an integrated part of the computing world.

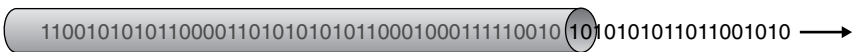


Figure 8-1 Today, encrypted binary data passes through network cables and airwaves.

Cryptography

Cryptography has had an interesting history and has undergone many changes through the centuries. It seems that keeping secrets has been important throughout the ages of civilization for one reason or another. Keeping secrets gives individuals or groups the ability to hide true intentions, gain a competitive edge, and reduce vulnerability.

The changes that cryptography has undergone throughout history closely follow the advances in technology. Cryptography methods began with a person carving messages into wood or stone, which were then passed to the intended individual who had the necessary means to decipher the messages. This is a long way from how cryptography is being used today. Cryptography that used to be carved into materials is now being inserted into streams of binary code that passes over network wires, Internet communication paths, and airwaves, as shown in Figure 8-1.

In the past, messengers were used as the transmission mechanism, and encryption helped protect the message in case the messenger was captured. Today, the transmission mechanism has changed from human beings to packets carrying 0's and 1's passing through network cables or open airwaves. The messages are still encrypted in case an intruder captures the transmission mechanism (the packets) as they travel along their paths.

History of Cryptography

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

Cryptography has roots that began around 2000 B.C. in Egypt when hieroglyphics were used to decorate tombs to tell the story of the life of the deceased. The practice was not as much to hide the messages themselves, but to make them seem more noble, ceremonial, and majestic. An illustration of hieroglyphics is shown in Figure 8-2.

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

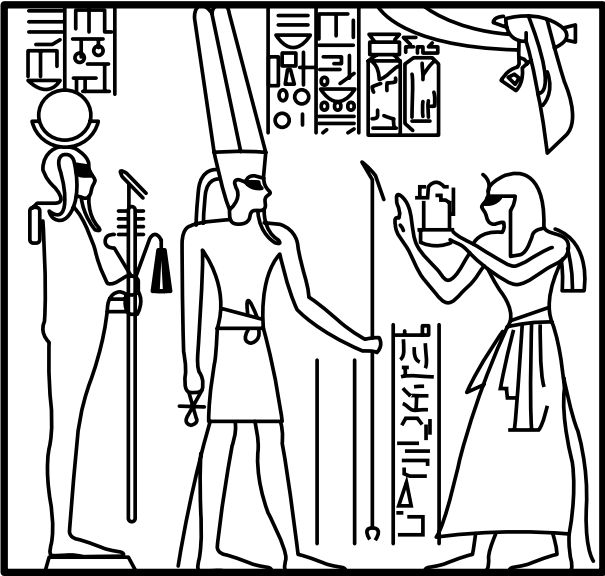


Figure 8-2 Hieroglyphics are the first recorded use of cryptography.

A Hebrew cryptographic method required the alphabet to be flipped so that each letter in the original alphabet is mapped to a different letter in the flipped alphabet. The encryption method was called *atbash*. An example of an encryption key used in the atbash encryption scheme is shown in following:

```

ABCDEFGHI JK LMNOPQ R STU VW XYZ
ZYXWVUTSR QP ONMLKJ I HGF ED CBA

```

For example, the word “security” is encrypted into “hvxfirgb.” What does “xrhhk” come out to be? This is a *substitution cipher*, because one character is replaced with another character. This type of substitution cipher is referred to as a *monoalphabetic substitution* because it uses only one alphabet, compared to other ciphers that use multiple alphabets at a time.

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 by writing a message on a sheet of papyrus, which was wrapped around a staff. (This would look like a piece of paper wrapped around a stick or wooden rod.) The message was only readable if it was around the correct staff, which allowed the letters to properly match

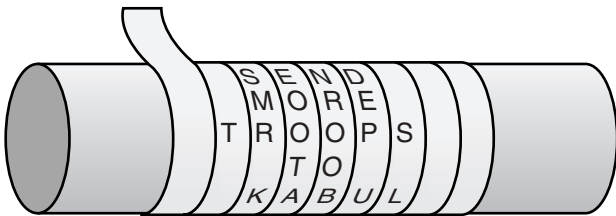


Figure 8-3 The scytale was used by the Spartans to decipher encrypted messages.

up. This is referred to as the *scytale* cipher, as shown in Figure 8-3. When the papyrus was removed from the staff, the writing appeared as just a bunch of random characters. The Greek government had carriers run these pieces of papyrus to different groups of soldiers. The soldiers would then wrap the papyrus around a staff of the right diameter and length and all the seemingly random letters would match up and form an understandable message. These could be used to instruct the soldiers on strategic moves and provide them with military directives.

In another time and place in history, Julius Caesar developed a simple method of shifting letters of the alphabet, similar to the atbash scheme. Today this technique seems too simplistic to be effective, but in that day not many people could read in the first place, so it provided a high level of protection. The evolution of cryptography continued as Europe refined its practices using new methods, tools, and practices throughout the Middle Ages, and by the late 1800s, cryptography was commonly used in the methods of communication between military factions.

During World War II, simplistic 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 history to date: Germany's *Enigma* machine. The Enigma machine had three rotors, a plugboard, and a reflecting rotor.

The originator of the message configured 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, which would substitute the original letter with a different letter. 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 advanced the rotors to a new setting. This process was done until the whole message was encrypted. Then the encrypted text was transmitted over the airwaves most likely to a 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 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, the Queen of Scots, lost her life in the sixteenth 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. The military has always had a big part in using cryptography by encoding information and attempting to decrypt their enemy's encrypted information. William Frederick Friedman published *The Index of Coincidence and Its Applications in Cryptography*. He is referred to as the "Father of Modern Cryptography" and broke many messages that were intercepted during WWII. Encryption has been used by many governments and militaries and has allowed great victory for some because of the covert maneuvers that could be accomplished in shrouded secrecy. It has also brought great defeat to others when their cryptosystems were discovered and deciphered.

As computers came to be, the possibilities for encryption methods and devices advanced, and cryptography efforts expanded exponentially. This era brought unprecedented opportunity for cryptographic designers and 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 come up with the U.S. Data Encryption Standard (DES). DES has been adopted as a federal government standard, is used worldwide for financial transactions, and is imbedded into numerous commercial applications. DES has had a rich history in computer-oriented encryption and has been in use for over 20 years.

Cryptography has had its days in the political limelight with governments enforcing transborder restrictions and hindering the use of cryptography in certain sectors by imposing export regulations. Law enforcement developed their own encryption chip, the *Clipper Chip*, to decipher communication that had to do with suspected criminal activity and drug movements, which has raised many questions about the public's privacy versus the government's right to eavesdrop. (These issues are addressed further in "The Government's Involvement in Cryptography" section.)

A majority of the protocols developed at the dawn of the computing age have had upgraded to include cryptography to add necessary layers of protection. Encryption is used in hardware devices and software to protect data, banking transactions, corporate extranets, e-mail, Web transactions, wireless communication, storing of confidential information, faxes, and phone calls.

The code breakers and cryptanalysis efforts and the amazing amount of 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 efforts and strategy. *Cryptanalysis* is a science of studying and breaking the secrecy of encryption algorithms and their necessary pieces. It is performed in academic settings and by curious and motivated hackers, either to quench their inquisitiveness or use their findings to commit fraud and destruction. Different types of cryptography have been used throughout civilization, but today it is deeply rooted in every part of our communication and computing world. Automated information systems and cryptography play a huge role in the effectiveness of militaries, functionality of governments, and 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

www.infosecuritymag.com/articles/july01/columns_logoff.shtml

<http://all.net/books/ip/Chap2-1.html>

www.cs.cornell.edu/Courses/cs513/2000 SP/L23.html

www.execpc.com/~alcourt/crypt.intro.html

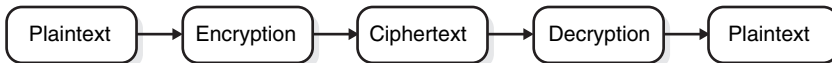
www.trincoll.edu/depts/cpsc/cryptography/index.html

http://dmoz.org/Science/Math/Applications/Communication_Theory/Cryptography/Historical/

Cryptography Definitions

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

Encryption is a method of transforming original data, called *plaintext* or *cleartext*, into a form that appears to be random and unreadable, which is called *ciphertext*. Plaintext is either in a form that can be understood 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 is stored on a computer, it is 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.



The process of encryption transforms plaintext into ciphertext and the process of decryption transforms ciphertext into plaintext.

A system 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 process will be. 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, as depicted in Figure 8-4.

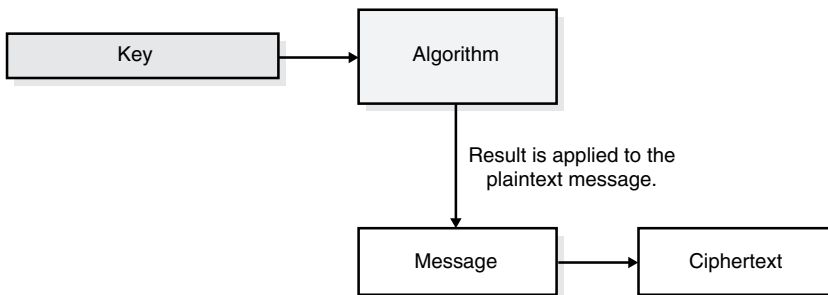


Figure 8-4 The key is inserted into the mathematical algorithm and the result is applied to the message, which ends up in ciphertext.

The *algorithm*, the set of mathematical rules, dictates how enciphering and deciphering take place. Many algorithms are publicly known and are not the secret part of the encryption process. The way that encryption algorithms work can be kept secret from the public, but many of them are publicly known and well understood. 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. The *key* can be any value that is made up of 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. The key is made up of random values within the keyspace range. 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.

A large keyspace allows for more possible keys. The encryption algorithm should use the entire keyspace and choose the values to make up the keys as random as possible. If a smaller keyspace were used, there would be fewer values to choose from when forming a key, as shown in Figure 8-5. This would increase an attacker's chance of figuring out the key value and deciphering the protected information.

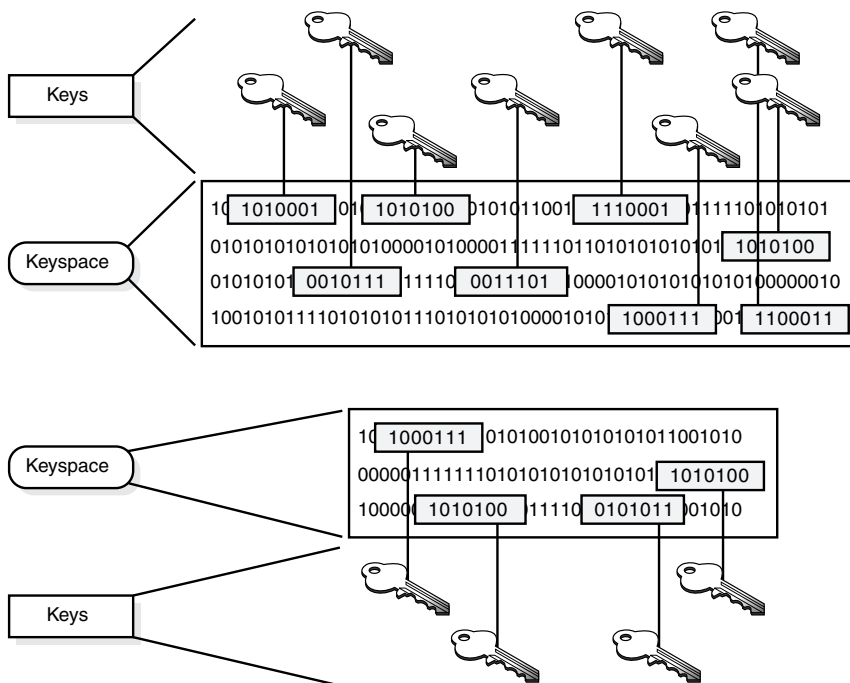


Figure 8-5 Larger keyspaces allow for more possible keys.

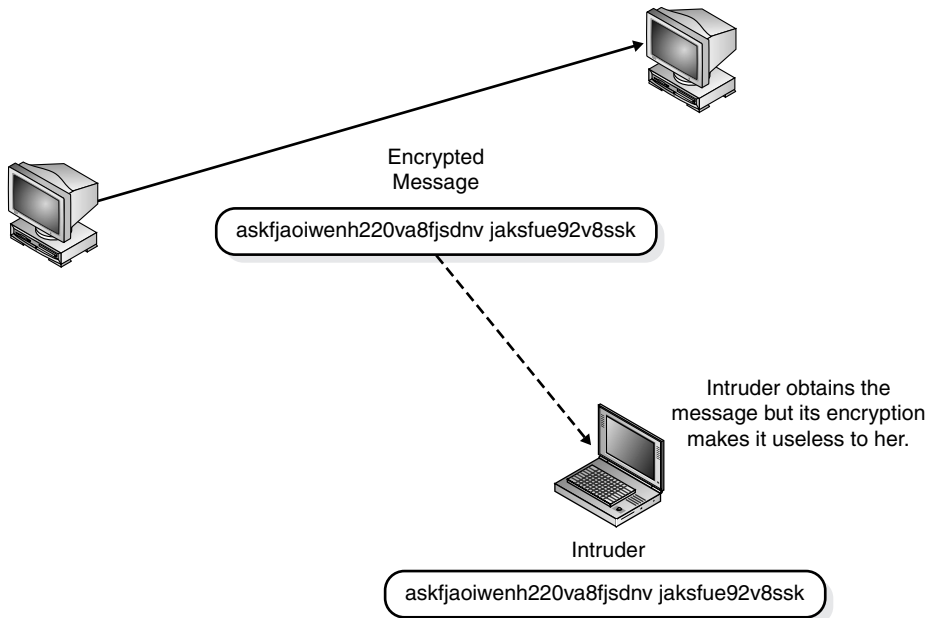


Figure 8-6 Without the right key, the captured message is useless to an attacker.

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 8-6.



NOTE Making and Breaking Encryption: Cryptography is the science of encrypting and decrypting written communication. It comes from the Greek word “kryptos,” meaning hidden, and “graphia,” meaning writing. Cryptography involves developing, testing, and studying the science of encryption methods.

Cryptanalysis is the process of trying to decrypt encrypted data without the key. When new algorithms are tested, they go through stringent processes of cryptanalysis to ensure that the new encryption process is unbreakable or that it takes too much time and resources to break.

Strength of the Cryptosystem

You are the weakest link. Goodbye!

The *strength* of the encryption method comes from the algorithm, secrecy of the key, length of the key, initialization vectors, and how they all work together. 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. Breaking a key has to do with processing an amazing number of possible values in the hopes of finding the one value that can be used to decrypt a specific message. The strength correlates to the amount of necessary processing power and time it takes to break the key or figure out the value of the key. Breaking a key 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 a very easy task or a task 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, and it takes 1.2 million years, then it is pretty darn strong.

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

The strength of the protection mechanism should be used in correlation to the sensitivity of the data being encrypted. It is not necessary to encrypt information about a friend's Saturday barbeque with a top secret NSA encryption algorithm, and it is not a good idea to send the intercepted KGB spy information using Pretty Good Privacy (PGP). Each type of encryption mechanism has its place and purpose.

Even if the algorithm is very complex and thorough, there are other issues within encryption that can weaken the strength of 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 strength. An extremely strong algorithm can be used, using a large keyspace, and a large and random key value, which are all the requirements for strong encryption, but if a user shares her key with others, these other pieces of the equation really don't matter.

An algorithm with no flaws, a large key, using all possible values within a keyspace, and protecting the actual key are important elements of encryption. If one is weak, it can prove to be the weak link that affects the whole process.

Goals of Cryptosystems

Cryptosystems can provide confidentiality, authenticity, integrity, and nonrepudiation services. It does not provide availability of data or systems. *Confidentiality* means that unauthorized parties cannot access information. *Authenticity* refers to validating the source of the message to ensure the sender is properly identified. *Integrity* provides assurance that the message was not modified during transmission, accidentally or intentionally. *Nonrepudiation* means that a sender cannot deny sending the message at a later date, and the receiver cannot deny receiving it. So if your boss sends you a message telling you that you will be receiving a raise that doubles your salary and it is encrypted, encryption methods can ensure that it really came from your boss, that someone did not alter it before it arrived to your computer, that no one else was able to read this message as it traveled over the network, and that your boss cannot deny sending the message later when he comes to his senses.

Different types of messages and transactions require a higher degree of one or all of the services that encryption 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 care more 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 institution world. Legal agencies may care more about the authenticity of messages that they receive. If information that was received ever needed to be presented in a court of law, its authenticity would certainly be questioned; therefore, the encryption method used should ensure authenticity, which confirms who sent the information.

Cryptography Definitions

- **Algorithm** Set of mathematical rules used in encryption and decryption
- **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 obtaining plaintext from ciphertext without a key or breaking the encryption



- **Cryptology** The study of both cryptography and cryptanalysis
- **Ciphertext** Data in encrypted or unreadable format
- **Encipher** Act of transforming data into an unreadable format
- **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** Possible values used to construct keys
- **Plaintext** Data in readable format, also referred to as cleartext
- **Work factor** Estimated time, effort, and resources necessary to break a cryptosystem



NOTE Repudiation: If David sends a message and then later claims that he did not send the message, this is an act of repudiation. When an encryption mechanism provides nonrepudiation, it means that the sender cannot deny sending the message and the receiver cannot deny receiving it. It's a way of keeping everybody honest.

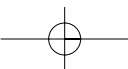
Types of Ciphers

There are two basic types of encryption ciphers: 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 moves the original text around. It rearranges the bits, characters, or blocks of characters to hide the original meaning.

Substitution Cipher

A substitution cipher uses a key to know 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. This is referred to as a shift alphabet.

If the Caesar Cipher is used with the English alphabet, when George wants to encrypt a message of "FBI," the encrypted message would be "IEL." Substitution is used in today's algorithms, but it is extremely complex compared to this example. Many differ-



ent types of substitutions take place usually with more than one alphabet. This example is only meant to show you the concept of how a substitution cipher works in its most simplistic form.

Transposition Cipher

In a *transposition cipher*, permutation is used, meaning that letters are scrambled. The key determines the positions that the characters are moved to, as illustrated in Figure 8-7.

This is a simplistic example of a transposition cipher and only shows one way of performing transposition. When introduced with complex mathematical functions, transpositions can become quite sophisticated and difficult to break. Most ciphers used today use long sequences of complicated substitutions and permutations together on messages. The key value is inputted into the algorithm and the result is the sequence of operations (substitutions and permutations) that are performed on the plaintext.

Simple substitution and transposition ciphers are vulnerable to attacks that perform *frequency analysis*. In every language, there are words and patterns that are used more often than others. For instance, in the English language, the words “the,” “and,” “that,” and “is” are very frequent patterns of letters used in messages and conversation. The beginning of messages usually starts “Hello” or “Dear” and ends with “Sincerely” or “Goodbye.” These patterns help attackers figure out the transformation between plaintext to ciphertext, which enables them to figure out the key that was used to perform the transformation. It is important for cryptosystems to not reveal these patterns.

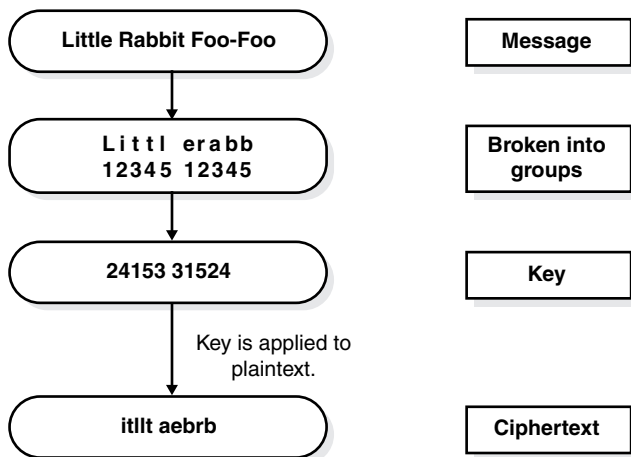


Figure 8-7 Transposition cipher

More complex algorithms usually use more than one alphabet for substitution and permutation, which reduces the vulnerability to frequency analysis. The more complicated the algorithm, the more the resulting text (ciphertext) differs from the plaintext; thus, the matching of these types of patterns becomes more difficult.

Running and Concealment Ciphers

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

More of the spy-novel-type ciphers would be 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 clever steps in the physical world around you. For instance, a key in this type of cipher could be a book page, line number, and word count. If I get a message from my super-secret spy buddy and the message reads "14967.29937.91158," this could mean for me to look at the first book in our predetermined series of books, the 49th page, 6th line down the page, and the 7th word in that line. So I write down this word, which is "cat." The second set of numbers start with 2, so I go to the 2nd book, 99th page, 3rd line down, and write down the 7th word on that line, which is "is." The last word I get from the 9th book in our series, the 11th page, 5th row, and 8th word in that row, which is "dead." So now I have come up with my important secret message, which is "cat is dead." 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. Another type of spy novel cipher is the *concealment cipher*. 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. So if 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 type 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 states that encryption and decryption will take place by choosing characters out of a predefined set of books. The key indicates the book, page, line, and word within that line. In substitution cipher, 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 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, say the predefined set of books, if he does not know the key, the message is still useless to him.

References

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

www.math.nmsu.edu/~crypto/Caesar.html

www.ssuet.edu.pk/taimoor/athar/ce-408/encryption/

<http://home.ecn.ab.ca/~jsavard/crypto/pp0102.htm>

Steganography

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

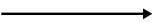
Steganography is a method of hiding data in another message so that the very existence of the data is concealed. Steganography is mainly used by hiding messages in graphic images. The least significant bit of each byte of the image can be replaced with bits of the secret message. This practice does not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

Steganography does not use algorithms or keys to encrypt information, but this is a process to hide data within another object so no one will detect its presence. A message can be hidden in a wave 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 in the hopes of detecting illegal copies of the images.

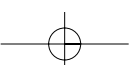


Steganography can hide a message inside a graphic.

Secret
Communist
Message
hidden in the
picture.



Weapons
hidden
under the
Kremlin.



References

www.jjtc.com/Steganography/

www.rit.edu/~vvr8205/crypto2/cryptopaper.html

The Government's Involvement with Cryptography

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

The government's cryptographic agency, the NSA, was granted the power to regulate the export of cryptographic mechanisms and equipment. This was done with the hopes of making encryption technology harder to obtain and be used by terrorists and criminals. Harry Truman created the NSA in 1952, and its main mission is to listen in on communications in the interest of national security for the United States. Its very existence is kept at 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 that enemies of the United States could not employ encryption methods that were too strong for it to break. These issues have caused tension and controversy between cryptography researchers, vendors, and the NSA pertaining to new cryptographic methods and the public use of them. The fear is that if the government controls all types of encryption and is allowed to listen in on private citizens' conversations, the obtained information may be misused in Big Brother ways. Also, if the government had the ability to listen in on everyone's conversations, there is little trust that this ability would not fall into the wrong hands for the wrong reasons.

Clipper Chip

In 1993, the government made a case that would enable them to place their own encryption chip, the *Clipper Chip*, in every American-made device that had a computer or computer components. This included telephones, TVs, personal computers, and more. The Clipper Chip was based on the SkipJack algorithm that was classified and never opened for public review or testing. A majority of the algorithms used today in cryptography have been publicly tested to ensure that the developers did not miss any important steps in building a complex and secure mechanism. Because the SkipJack

algorithm was not open for public review, many people in the public do not trust its effectiveness.

The Clipper Chip was a NSA-designed tamperproof chip for encrypting data. It is one of the two chips implemented in the U.S. government's Escrowed Encryption Standard (EES). Each chip has a unit key, which is used to encrypt a copy of each user's session key, not the message itself. Each Clipper Chip has a unique serial number and a copy of the unit key is stored in the database under this serial number. The sending Clipper Chip generates and sends a Law Enforcement Access Field (LEAF) value included in the transmitted message. This field value contains the serial number of the Clipper Chip used to encrypt the message in the first place. This is how the government, or law enforcement, knows which unit key to retrieve from the database. This unit key enables them to decrypt and find out the session key, which enables them to actually decrypt the message and eavesdrop on the conversation, as shown in Figure 8-8. The unit key is split into two pieces and kept in different databases maintained by two different escrow agencies.

There was quite a public outcry pertaining to the Clipper Chip and its threat on personal privacy. Eventually, the government stopped supporting the Clipper Chip and most companies turned to software-based encryption programs instead of an actual hardware chip. Several deficiencies were found in the Clipper Chip and because it was viewed as being very invasive to the public's privacy, it quickly lost support and was dropped. The Clipper Chip initiative was abandoned and replaced with policies aimed at controlling the proliferation and use of cryptography in the public. It was just too Big Brother for society.

Some Weaknesses Found in the Clipper Chip

- The SkipJack algorithm was never publicly scrutinized and tested.
 - An 80-bit key is very weak.
 - A 16-bit checksum can be defeated.
 - The Clipper Chip ID tagged and identified every communication session.
-

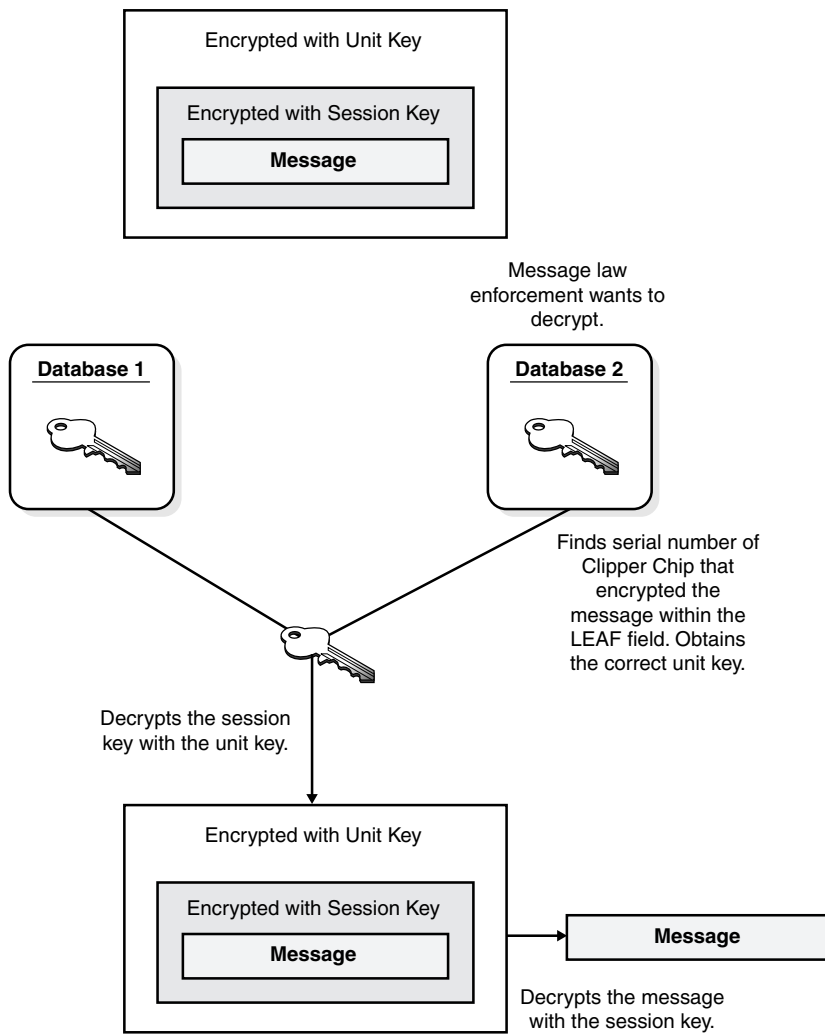


Figure 8-8 The unit key decrypts the session key.

References

- www.swiss.ai.mit.edu/6095/articles/froomkin-metaphor/partIC.html#ToC27
- <http://cse.stanford.edu/class/cs201/current/Projects/clipper-chip/debate.html>
- www.rsa.com/rsalabs/faq/6-2-4.html

Key Escrow

In many cases, it is necessary for law enforcement to have access to certain conversations to try and learn about possible terrorist attacks, drug deals, or murder attempts. However, there is always a tricky balance between this activity and a citizen's right to private communication. There needs to be a proper system of checks and balances in place to ensure that abuse of this type of power is not taking place. This is where key escrow comes in. The unit keys are split into two sections and are given to two different escrow agencies to maintain. This is done so that one agency does not have the ability to abuse this technology by itself and that three entities are involved with decrypting this type of data: two agencies and a law enforcement representative.

For an officer to access data that is encrypted, he must get a court order to request the unit key in the first place. The officer submits this court order to both escrow agencies, which in turn release the key sections, as shown in Figure 8-9. The sections are combined into a full unit key, and it is used on only the specified data that is outlined in the court order. This is all outlined in the U.S. EES.

The Clipper Chip uses the concept of key escrow and key recovery. A key escrow implementation can also be used in software using public key cryptography. In these cases, the public key is available to encrypt and decrypt messages, but the private key is split up into two or more pieces and stored by different entities. When it is necessary to decrypt information, say during a wiretap, then each entity must supply its piece of the private key, which will be combined to create one useful key. This provides another layer of protection because two or more people would have to supply part of the private key to decrypt information. This is referred to as the *fair cryptosystems*, which uses software instead of hardware chips. The algorithm in fair cryptosystems does not need to be secret; thus, well-tested and well-known algorithms and protocols can be used.

References

www.itl.nist.gov/fipspubs/fip185.htm

<http://security.isu.edu/isl/fips185.html>

www.cs.georgetown.edu/~denning/crypto/Lathe-Gambit.txt

www.sims.berkeley.edu/courses/is224/s99/GroupC/pr2/s4.html

Methods of Encryption

Although there can be several pieces to an encryption method, the two main pieces are the algorithms and the keys. As stated earlier, algorithms are usually complex

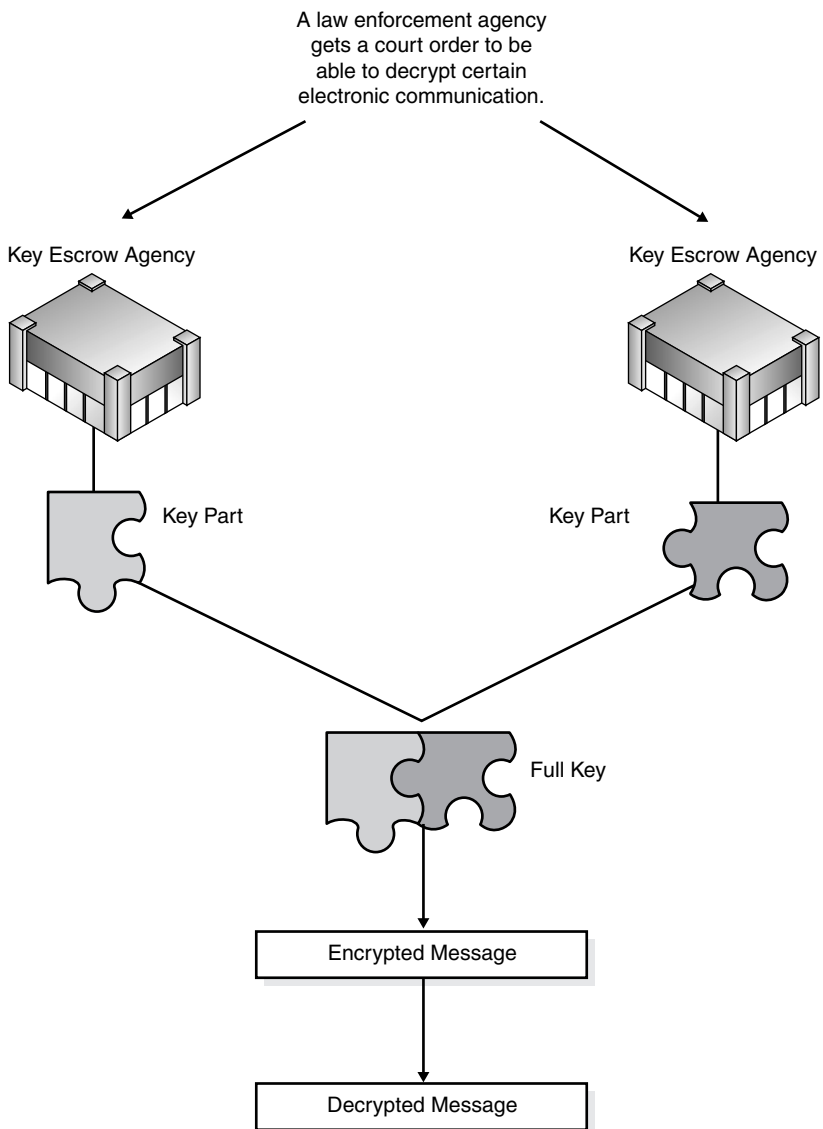


Figure 8-9 In a key escrow arrangement, the key necessary to decrypt traffic is split and kept by at least two different parties.

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 inserted into the algorithm. 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 methods, the receiver and the

sender use the same key and in other encryption methods, they must use different keys for encryption and decryption purposes. The following sections explain the difference between these two types of encryption methods.

Symmetric versus Asymmetric Algorithms

Cryptography algorithms use either *symmetric keys*, also called secret keys, or *asymmetric keys*, also called public keys. As if encryption was not complicated enough, the titles that are used to describe the key types only make it worse. Just pay close attention and we will get through this just fine.

Symmetric Cryptography

In a cryptosystem that uses symmetric cryptography, both parties will be using the same key for encryption and decryption, as shown in Figure 8-10. This provides dual functionality. As we said, 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 this key got into an intruder's hand, that intruder would have the ability to decrypt any intercepted message encrypted with this key.

Each pair of users who want to exchange data using symmetric key encryption must have their own set of keys. This means 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

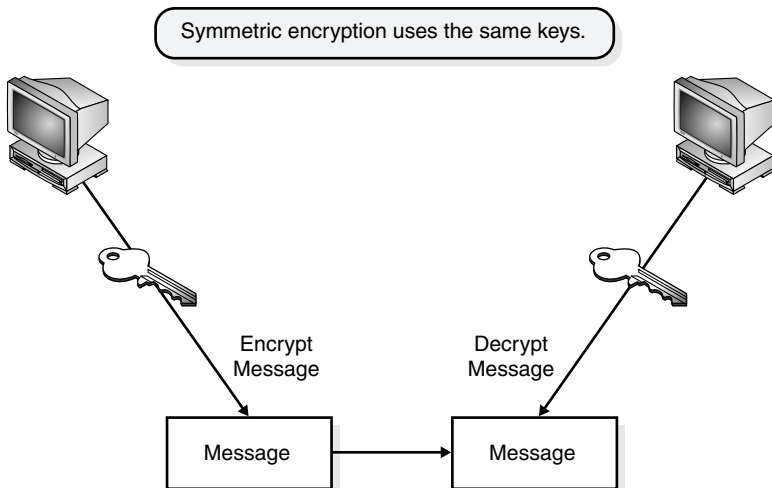


Figure 8-10 When using symmetric algorithms, the sender and receiver use the same key for encryption and decryption functions.

encryption with Norm and Dave, he now 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 very daunting task. If Dan were going to communicate with 10 other people, then he would need to keep track of 45 different keys. If Dan were going to communicate with 100 other people, then he would have to maintain and keep up with 4,950 symmetric keys. Dan is a pretty bright guy, but does not necessarily want to spend his days looking for the right key to be able to communicate with Dave.

The security of the symmetric encryption method is completely dependent on how well users protect the key. This should raise red flags to 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 to Norm for the first time, Dan has to figure out how to get Norm the right key. It is not safe to just send it in an e-mail message because the key is not protected and it can be easily intercepted and used by attackers. Dan has to get the key to Norm through an *out-of-band method*. Dan can save the key on a floppy disk and walk over to Norm's desk, send it to him via snail mail, or have a secure carrier deliver it to Norm. This is a huge hassle, and each method is very clumsy and insecure.

Because both users use 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 who actually sent a message if two people are using the exact same key.

Well, if symmetric cryptosystems have so many problems and flaws, why use them at all? They are very fast and can be hard to break. Compared to asymmetric systems, symmetric algorithms scream in speed. They can encrypt and decrypt large amounts of data that would take an unacceptable amount of time if an asymmetric algorithm was used instead. It is also very difficult to uncover data that is encrypted with a symmetric algorithm if a large key size was used.

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

- **Key distribution** It requires a secure mechanism to deliver keys properly.

- **Scalability** Each pair of users needs a unique pair of keys, so the number of keys grow exponentially.
- **Limited security** It can provide confidentiality, but not authenticity or nonrepudiation.

The following are examples of symmetric key cryptography algorithms and will be explained in the “Stream and Block Ciphers” section:

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

References

<http://csrc.nist.gov/publications/nistpubs/800-7/node208.html>

<http://developer.netscape.com/docs/manuals/security/pkin/contents.htm>

www1.tepkom.ru/users/ant/Articles/Pkcstane.html

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 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 only be known to 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 8-11 illustrates an asymmetric cryptosystem.

The public and private keys are mathematically related, but cannot be derived from each other. This means that if an evildoer gets a copy of Bob’s public key, it does not mean he can now use some mathematical magic and find out Bob’s private key.

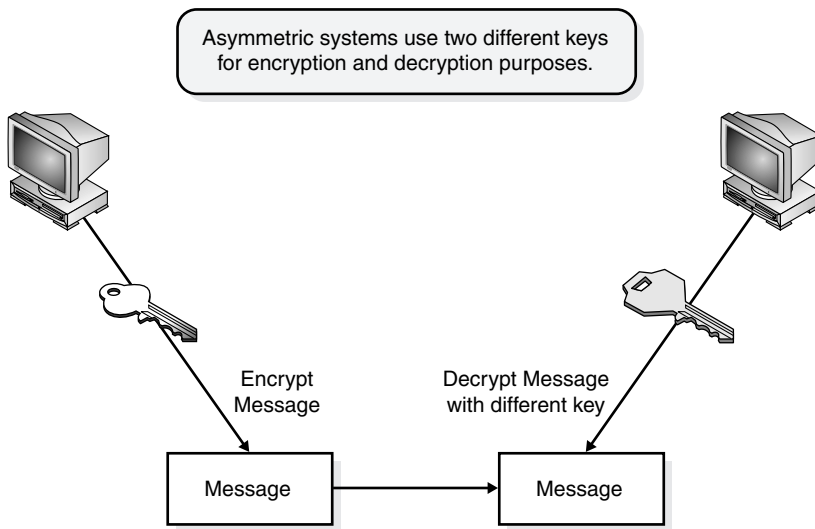


Figure 8-11 Asymmetric cryptosystem

If Bob encrypts a message 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 back 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 exact same key when using an asymmetric key encryption technology.

Bob can encrypt a message 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 that the message really came from Bob. A message can only be decrypted with a public key 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. When 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.

Now the receiver can also encrypt her response with her private key instead of using Bob's public key. Why would she do that? She wants Bob to know that the message came from her and no one else. If she encrypted the response with Bob's public key, it does not provide authenticity because anyone can get a hold of Bob's public key. If she

uses her private key to encrypt the message, then Bob can be sure that 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 that the message originated from a specific entity.

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 message with her private key. This provides assurance to the receiver that the only person who could have encrypted the message is the individual who has possession of that private key. If the sender encrypted the message with the receiver's public key, authentication is not provided because this public key is available to anyone.

Encrypting a message 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; thus, confidentiality is not ensured.

For a message to be in a *secure and signed format*, the sender would encrypt the message with her private key and then encrypt it again with the receiver's public key. The receiver would then need to decrypt the message with his own private key and then decrypt it again with the sender's public key. This provides confidentiality and authentication for that delivered message. The different encryption methods are shown in Figure 8-12.

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 is encrypted with a private key, it cannot be decrypted with a private key. If data is encrypted with a private key, it must be decrypted with the corresponding public key. If data is encrypted with a public key, it must be decrypted with the corresponding private key. Figure 8-13 further explains the steps of a signed and secure message.

An asymmetric cryptosystem works much slower than symmetric systems, but can provide confidentiality, authentication, and nonrepudiation depending on its configuration and use. 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 "Public Key Cryptography" section will show how these two systems can be used together to get the best of both worlds.

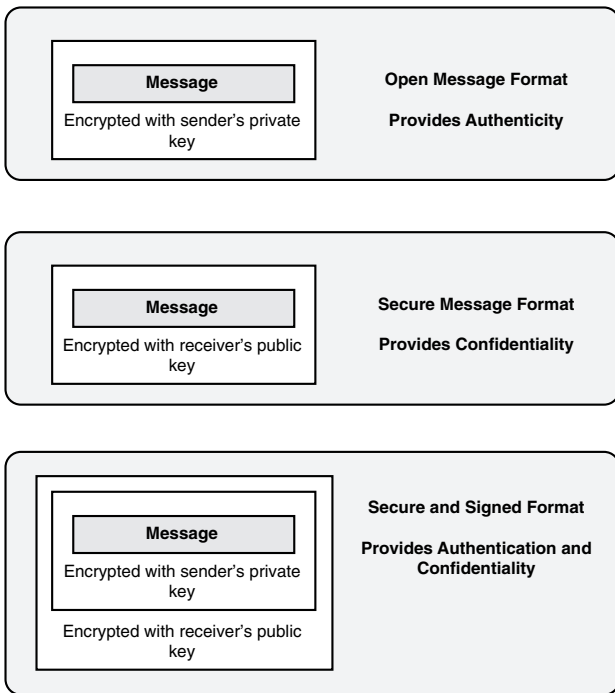


Figure 8-12 The way that the sender encrypts the message dictates the type of security service that will be provided.

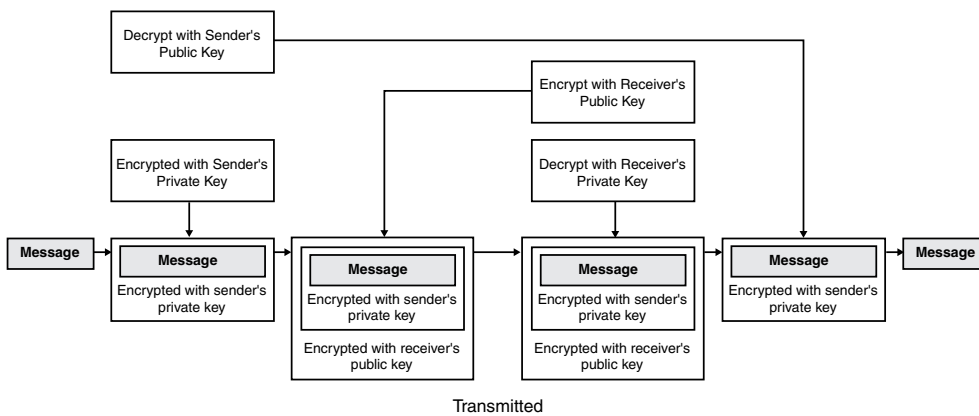


Figure 8-13 A secured and signed message is encrypted twice with the sender and the receiver's keys.

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

- Strengths
 - Better key distribution than symmetric systems
 - Better scalability than symmetric systems
 - Can provide confidentiality, authentication, and nonrepudiation
- Weaknesses
 - Works much slower than symmetric systems

The following are examples of asymmetric key algorithms:

- RSA
- Elliptic Curve Cryptosystem (ECC)
- Diffie-Hellman
- El Gamal
- Digital Signature Standard (DSS)

(These will be explained further in the sections under the “Asymmetric Encryption Algorithms” heading later in the chapter.)

References

<http://csrc.nist.gov/publications/nistpubs/800-7/node210.html>

www.eco.utexas.edu/~norman/BUS.FOR/course.mat/SSim/history.html

www.maths.mq.edu.au/~steffen/old/PCry/report/node8.html

www.emporia.co.za/TechnicalEncryption.asp

Stream and Block Ciphers

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

There are two main types of symmetric algorithms: stream and block ciphers. Like their names sound, block ciphers work on blocks of plaintext and ciphertext, whereas stream ciphers work on streams of plaintext and ciphertext, one bit or byte at a time.



Block Cipher

When a *block cipher* algorithm is used for encryption and decryption purposes, the message is divided into blocks of bits. These blocks are then put through substitution, transposition, and other mathematical functions. The algorithm dictates all the possible functions available to be used on the message, and it is the key that will determine what order these functions will take place. Strong algorithms make reengineering, or trying to figure out all the functions that took place on the message, basically impossible.

It has been said that the properties of a cipher should contain confusion and diffusion. Different unknown key values cause confusion, because the attacker does not know these values, and diffusion is accomplished by putting the bits within the plaintext through many different functions so that they are dispersed throughout the algorithm. An analogy is an example where Dusty was given the task of finding 25 people. The 25 people start off as one group in his living room and have their own map that lays out paths to their destinations. Each person has a destination of a particular city in a different state. The 25 people disperse and reach their destinations and it is up to Dusty to find them. Because he does not have a copy of each and every person's map (or their keys), it brings confusion to the game. Because each person is in a different state throughout the United States, it brings along diffusion.

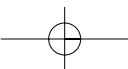
Block ciphers use diffusion and confusion in their methods. Figure 8-14 shows a simple block cipher. It has 16 inputs and each input represents a bit. This block cipher has two layers of 4-bit substitution boxes called *S-boxes*. Each S-box contains a lookup table that instructs how the bits should be permuted or moved around. The key that is used in the encryption process dictates what S-boxes are used and in what order.

Figure 8-14 shows that the key dictates what S-boxes are to be used when scrambling the original message from readable plaintext to encrypted nonreadable ciphertext. Each S-box can have different types of functions, mathematical formulas, and methods to be performed on each particular bit. The key provides the confusion because the attacker would not know which S-boxes would be used during the encryption process and all the permutations that happen on the bits is the diffusion, because they are moved between different S-boxes and put through different steps of scrambling. In this example, only two rounds are performed on the message.

This example is very simplistic—most block ciphers work with blocks of 64 bits and many more S-boxes are usually involved. Strong and efficient block cryptosystems use random key values so an attacker cannot find a pattern as to which S-boxes are chosen and used.

Stream Cipher

As stated earlier, a block cipher performs mathematical functions on blocks of data. A stream cipher does not divide a message up into blocks; instead, a stream cipher treats



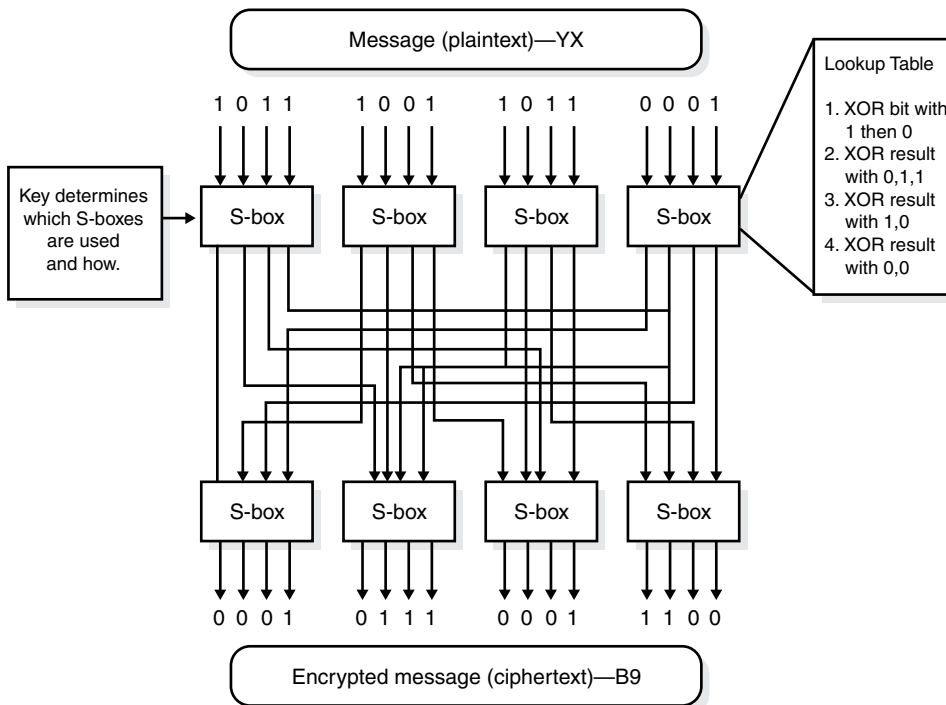


Figure 8-14 In block cipher algorithms, a message is divided into blocks of bits and mathematical functions are performed on those blocks.

the message as a stream of bits or bytes and performs mathematical functions on them individually.

When using a stream cipher, the same plaintext bit or byte will be transformed into a different ciphertext bit or byte each time it is encrypted. Some stream ciphers use a *keystream generator*, which produces a stream of bits that is XORed with the plaintext bits to produce ciphertext, as shown in Figure 8-15. (XOR stands for exclusive OR.)



NOTE Exclusive OR (XOR) Functionality: XOR is an operation that is applied to two bits. It is a function in binary mathematics. If both bits are the same, the result is zero ($1 + 1 = 0$). If the bits are different than each other, the result is one ($1 + 0 = 1$).

Example:

Message stream	1001010111
Keystream	0011101010
Ciphertext stream	1010111101

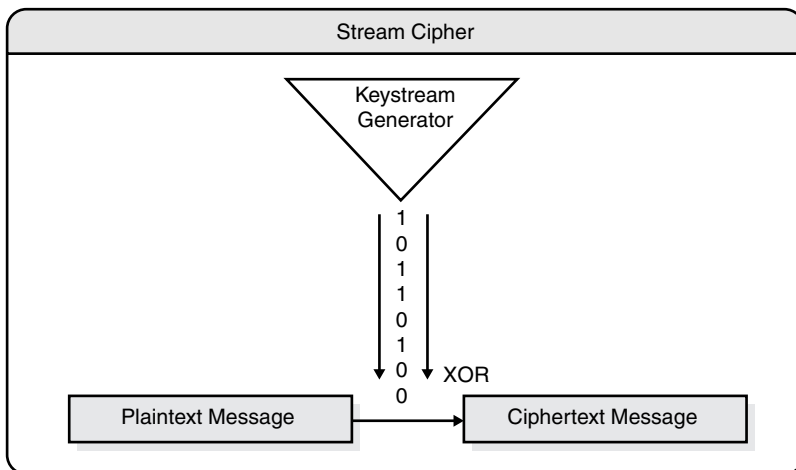


Figure 8-15 The value that is generated by the keystream generator is XORed with the bits of the plaintext message.

If the cryptosystem was only dependent upon this keystream generator, 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. It is the key that provides the randomness of the encryption process. As stated earlier, most encryption algorithms are public so people know how they work. So the secret to the secret sauce is the key. In stream ciphers, the key also provides randomness, but to the keystream that is actually applied to the plaintext. The key is a random value input into the stream cipher, which it uses to ensure the randomness of the keystream data. This concept is shown in Figure 8-16.

A strong and effective stream cipher algorithm contains the following characteristics:

- Long periods of no repeating patterns within keystream values.
- Statistically unpredictable.
- The keystream is not linearly related to the key.
- Statistically unbiased keystream (as many 0's as 1's).

Because stream ciphers encrypt and decrypt one bit at a time, they are more suitable for hardware implementations. Block ciphers are easier to implement in software because they work with blocks of data that the software is used to working with, which

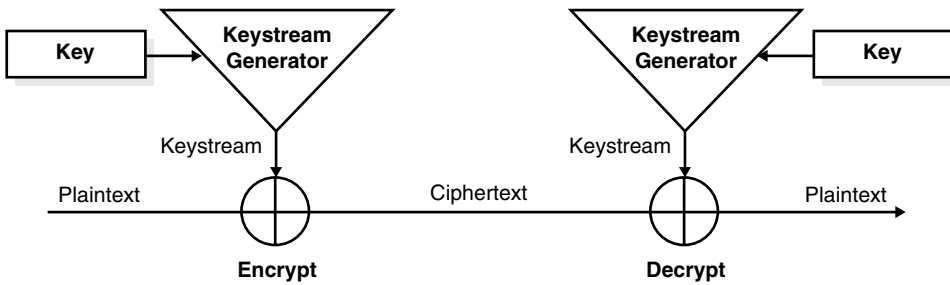


Figure 8-16 A keystream and key are needed for the encryption and the decryption process.

is usually the width of a data bus (64 bits). Stream ciphers are intensive because each bit must be manipulated, which works better at the silicon level. To make things just a little more confusing, block ciphers sometimes work in a mode that emulates a stream cipher. No one said cryptography was easy and we have not even touched any of the mathematics involved!

Types of Symmetric Systems

There are several types of symmetric algorithms 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 two identical keys are used to encrypt and decrypt the data.

Data Encryption Standard (DES)

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 computer systems during the 1960s and initiated a cryptography program in the early 1970s. NIST invited vendors to submit data encryption techniques to be used as a public cryptographic standard. IBM had been developing encryption algorithms to protect financial transactions. In 1974, IBM's 128-bit algorithm, named *Lucifer*, was submitted and accepted. There was controversy about if the NSA weakened Lucifer on purpose to give the agency the ability to decrypt messages not intended for them, but in the end, Lucifer became a national cryptographic standard in 1977 and an American National Standards Institute (ANSI) standard in 1978.

DES has been implemented in a majority of commercial products using cryptography functionality and in 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 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 of the 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 researches disagreed, but DSA wanted to move on to a newer, more secure, and less popular algorithm as the new standard.

NSA's decision caused major concern and negative feedback pertaining to dropping its support for DES. At that time, it was shown that DES still provided the necessary level of protection, it would take a computer thousands of years to crack it, it was already embedded in thousands of products, and 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, which broke DES in three days. It contained 1,536 microprocessors running at 40 MHz, 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 only become more feasible for the average attacker. This brought around 3DES, which provides stronger protection. 3DES performs encryption, decryption, and then encryption on a message with independent keys.

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 and will most likely be accepted and widely used in the public arena.

How Does DES Work?

How does DES work again? Answer: Voodoo magic and a dead chicken.

DES is a 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 8 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. A block is made up of 64 bits and is divided in half and each character is encrypted one at a time. The characters 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 that is inputted into the algorithm. The result is a 64-bit block of ciphertext.

There are several modes of operations when using block ciphers. 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 in a different environment with totally different types of requirements. It is important that vendors who employ DES understand the different modes and which one to use for which purpose.

DES has four distinct modes of operation that are used in different situations for different types of results.

Electronic Code Book (ECB) Mode This mode is the native encryption method for DES and 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. This mode is usually used for small amounts of data like encrypting and protecting encryption keys.

Every key has a different code book. The code book provides the recipe of substitutions and permutations that will be performed on the block of plaintext. Because this mode works with blocks of data independently, data within a file does 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 independent of any other table or record. Other DES modes are dependent upon the text that was 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.

This mode is used for challenge-response operations and some key management tasks. It is also used to encrypt personal identification numbers (PINs) in ATM machines for financial institutions. It is not used to encrypt large amounts of data because patterns would eventually show themselves.

Cipher Block Chaining (CBC) Mode In ECB mode, a block of plaintext and a key will always give the same ciphertext. This means that if the word "balloon" was encrypted and the resulting ciphertext was "hwicssn" each time it was encrypted using the same key, the same ciphertext would always be given. This can show evidence of a pattern, which if an evildoer put some effort into revealing, could get him 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 is processed in the algorithm and applied to the next block of text, as shown in Figure 8-17. This gives a more random resulting ciphertext. A value is extracted and used from the previous block of text. This provides dependence between the blocks and in a

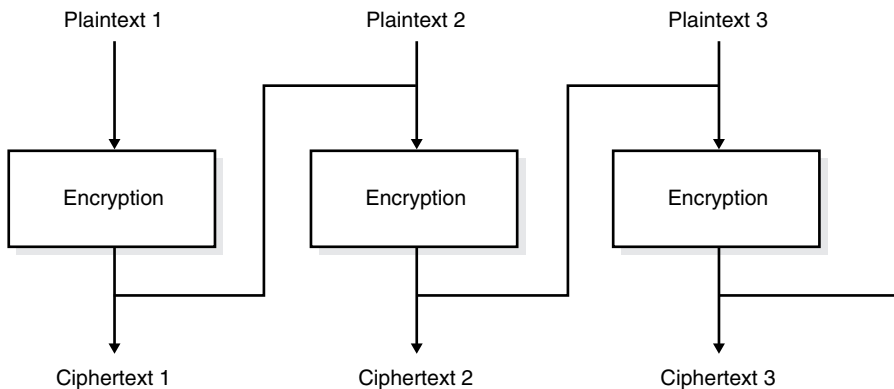


Figure 8-17 In CBC mode, ciphertext from the pervious block of data is used in encrypting the next block of data.

sense they are chained together. This is where the title of Cipher Block Chaining (CBC) comes from, and it is this chaining effect that hides any repeated patterns.

The results of one block are fed into the next block, meaning that 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.

Cipher Feedback (CFB) Mode In this mode, the previously generated ciphertext from the last encrypted block of data is inputted into the algorithm to generate random values. These random values are processed with the current block of plaintext to create ciphertext. This is another way of chaining blocks of text together, but instead of using a value from the last data block, CFB mode uses the previous data block in the ciphertext and runs it through a function and combines it with the next block in line. This mode is used when encrypting individual characters is required.

Output Feedback (OFB) Mode This mode is very similar to *Cipher Feedback (CFB)* mode, but if DES is working in *Output Feedback (OFB)* mode, it is functioning like a stream cipher by generating a stream of random binary bits to be combined with the plaintext to create ciphertext. The ciphertext is fed back to the algorithm to form a portion of the next input to encrypt the next stream of bits.

As previously stated, block cipher works on blocks of data and stream ciphers work on a stream of data. Stream ciphers use a keystream method of applying randomization and encryption to the text, whereas block ciphers use an S-box-type method. In OFB

mode, the DES block cipher crosses the line between block cipher and stream cipher and uses a keystream for encryption and decryption purposes.



NOTE Do not get flustered with the complexity of these different DES modes. As of this writing, the CISSP test does not delve that deeply into the intricacies of algorithms. These modes are good to know and understand, but if all of these concepts are new to you, focus on the main components of DES.

References

www.rsa.com/rsalabs/faq/3-2.html

<http://axion.physics.ubc.ca/crypt.html>

www.cryptography.com/

Triple-DES (3DES)

We went from DES to Triple-DES (3DES), so it might seem that we skipped Double-DES. We did. Double-DES has a key length of 112 bits, but its work factor is about the same as DES; thus, it is no more secure than DES. So we will 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. Many financial and banking applications have incorporated 3DES.

3DES uses 48 rounds in its computation, which makes it highly resistant to differential cryptanalysis and approximately 2^{56} times stronger than DES. However, because of the extra work that 3DES performs, there is a heavy performance hit and 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 to come.

Advanced Encryption Standard (AES)

After DES was used as an encryption standard for over 20 years and it was able to be cracked in a relative short amount of time, NIST decided a new standard, the Advanced Encryption Standard (AES), needed to be put into place. This decision was announced in January 1997, and a request for AES candidates was made. The AES was to be a

symmetric block cipher algorithm supporting keys sizes of 128-, 192-, and 256-bit keys. The following five algorithms were the finalists:

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

Rijndael was the NIST's choice in replacing DES. It is now the algorithm that is required to protect sensitive, but unclassified, U.S. government information. Rijndael is a block cipher with a variable block length and key length.

IDEA

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

The IDEA algorithm offers different modes similar to the modes described in the DES section, but it is much harder to break than DES. IDEA is used in the PGP encryption software. It was thought to replace DES, but it is patented, meaning that licensing fees would have to be paid to use it.

Blowfish

Blowfish is a block cipher that works on 64-bit blocks of data. The key length can be up to 448 bits and the data blocks go through 16 rounds of cryptographic functions. Bruce Schneier designed it.

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 usually 32, 64, or 128 bits and the key size goes up to 2,048 bits. RC5 was patented by RSA Data Security in 1997.

Asymmetric Encryption Algorithms

There are several types of asymmetric algorithms used in the computing world today. They may have different internal mechanisms and methods, but the one thing they do have in common is that they are all asymmetric. This means that a different key is used to encrypt a message than the key that is used to decrypt a message.

RSA

RSA, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman, is a public key algorithm that is the most understood, easiest to implement, and most popular when it comes to asymmetric algorithms. *RSA* is a worldwide de facto standard and can be used for digital signatures and encryption. It was developed in 1978 at MIT and provides authentication as well as 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 activities required to decrypt a message from ciphertext to plaintext using a public key is comparable to factoring the product of two prime numbers. (A prime number is a positive whole number with no proper divisors, meaning the only numbers that can divide a prime number is one 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 is used in many Web browsers with the Secure Sockets Layer (SSL) protocol. PGP and government systems that use public key cryptosystems (encryption systems that use asymmetric algorithms) also use *RSA*.

El Gamal

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

Elliptic Curve Cryptosystems (ECCs)

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 that *RSA* provides: digital signatures, secure key distribution, and encryption. One differing factor is *ECC*'s efficiency. Some devices have limited processing capacity, storage, power supply, and bandwidth like the newer 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 required by RSA and other algorithms, so it is used in these types of devices.

In most cases, the longer the key length, the more protection that is provided, but ECC can provide the same level of protection with a key size that is smaller 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.

ECC cryptosystems use the properties of elliptic curves in their public key systems. The elliptic curves provide ways of constructing groups of elements and specific rules of how the elements within these groups combine. The properties between the groups are used to build cryptographic algorithms.

References

www.cs.berkeley.edu/~daw/crypto.html

<http://csrc.nist.gov/encryption/aes/>

www.rsa.com/rsalabs/faq/3-6-8.html

www.sans.org/infosecFAQ/encryption/blowfish.htm

www.cryptoman.com/elliptic.htm

Hybrid Encryption Methods

Comparisons were made between symmetric and asymmetric algorithms earlier. So because symmetric has some downfalls, surely asymmetric will be our saving grace, right? Not so fast.

Asymmetric does provide more security services than the symmetric methods. It provides confidentiality by encryption. It also provides nonrepudiation when a sender encrypts a message using his private key, it provides integrity because if the message was tampered with it could not be properly decrypted, and it provides access control because only the people with the private key and corresponding public key can access the encoded data. However, asymmetric algorithms are unacceptably slow. Their algorithms are so complex and intensive that they require more system resources and take too long to encrypt and decrypt messages.

We just can't seem to win. So we turn to a hybrid system that uses symmetric and asymmetric encryption methods together and call it public key cryptography.

Public Key Cryptography

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. It is a hybrid use of two different algorithms: asymmetric and symmetric. Each algorithm has its pros and cons, so using them together can bring together the best of both worlds.

How Does Public Key Cryptography Work?

We have established that symmetric cryptography provides limited security because two users use the same key, and although asymmetric cryptography enables the two users to use different keys, it is too slow when compared to symmetric methods. So some really smart people decided to use them together to accomplish a high level of security in an acceptable amount of time.

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

When a secret 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 key to use to decrypt the message. You do not want this key to travel unprotected, because if the message was intercepted and the key was not protected, an evildoer could intercept the message that contains the necessary key to decrypt your message and read your information. If the secret key that is needed to decrypt your message is not protected, then there is no use in encrypting the message in the first place. So we use an asymmetric algorithm to encrypt the secret key, as depicted in Figure 8-18. Why do we use the symmetric algorithm on the message and the asymmetric algorithm on the key? We said earlier that 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).

So 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 secret key. The key needs to be protected so Bill encrypts the secret key with an asymmetric key. Remember that asymmetric algorithms use private and public keys, so Bill will encrypt the secret key Paul needs to perform decryption with Paul's public key. Now Bill has ciphertext from the message and ciphertext from the secret key. Why did Bill encrypt the secret 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

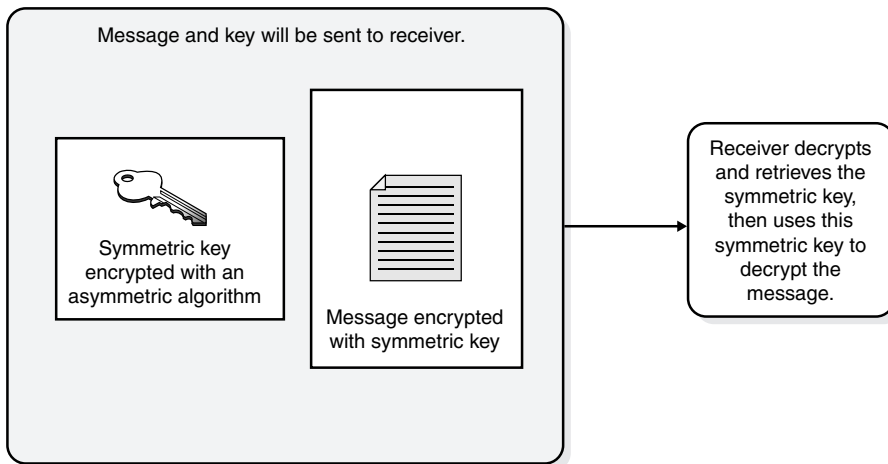
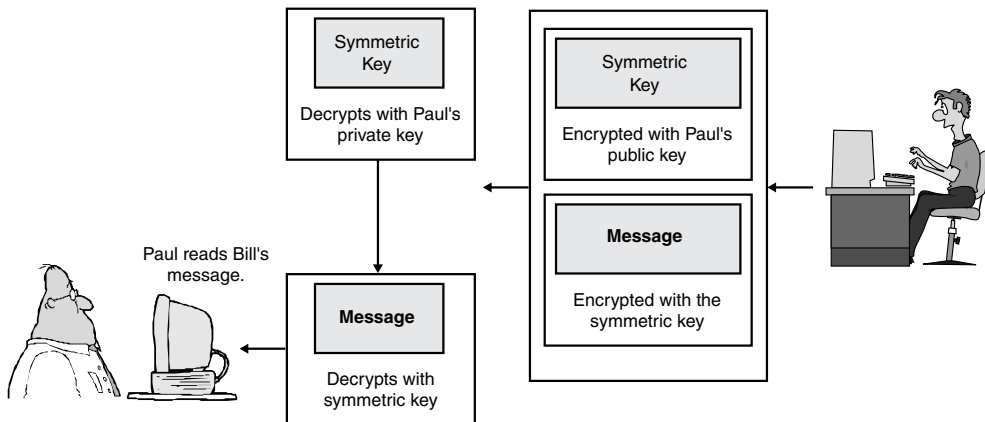


Figure 8-18 In a hybrid system, the asymmetric key is used to encrypt the secret key and the secret key is used to encrypt the message.

Bill's public key could decrypt it and retrieve the secret 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 secret key with Paul's public key. If Paul has done a good job protecting his private key, then he will be the only one who can read Bill's message.



Bill uses public key cryptography to send Paul a message.

So Paul receives Bill's message and Paul uses his private key to decrypt the secret key. Paul then uses the secret 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 go 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, 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:

- Asymmetric algorithm performs encryption and decryption by using public and private keys.
- Symmetric algorithm performs encryption and decryption by using a secret key.
- A secret key is used to encrypt the actual message.
- Public and private keys are used to encrypt the secret key.
- A secret key is synonymous to a symmetric key.
- An asymmetric key refers to a public or private key.



NOTE **Diffie-Hellman Key Exchange:** In 1976, Dr. W. Diffie and Dr. M.E. Hellman performed open research in cryptography and were the first to introduce the notion of public key cryptography. This notion allowed users to handle key distribution electronically in a secure fashion. This evolved into the Diffie-Hellman key exchange.

This method of key exchange enables users to exchange secret keys over a nonsecure medium. The Diffie-Hellman algorithm is used for key distribution and it cannot be used to encrypt and decrypt messages.

This means that Dr. Diffie and Dr. Hellman came up with the whole public key/private key concept.



NOTE **Public versus Private Key Cryptography:** It can get confusing when one is trying to learn the concepts of cryptography and unfortunately, the naming scheme does not always help out.

When the term “public key cryptography” is used, it is describing a system that uses an asymmetric algorithm that encrypts the secret keys. This system employs public and private keys. A sender might encrypt a message with the receiver's public key, and the receiver must decrypt it with her private key.

When the term “private key cryptography” is used, it is describing a system that is using a symmetric algorithm; thus, the sender and receiver use the same key for encryption and decryption purposes.

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 (either public or private key) encrypts the secret key. Table 8-1 outlines the differences between symmetric and asymmetric algorithms.

Session Keys

A *session key* is a secret key that is used to encrypt messages between two users. A session key is not any different than the secret key that was described in the previous section, but it is only good for one communication session between users.

If Tanya had a secret key she used to encrypt messages between Lance and herself all the time, then this secret key would not be regenerated or changed. They would use the exact same key each and every time they communicated using encryption. However, using the same key over and over again increases the chances of the key being captured and the secure communication being compromised. If, on the other hand, a new secret key was generated each time Lance and Tanya wanted to communicate, as shown in Figure 8-19, it would only be used during their one dialog and then destroyed. If they wanted to communicate an hour later, a new session key would be created and shared.

Table 8-1 Different Characteristics Between Symmetric and Asymmetric Systems

Attributes	Symmetric	Asymmetric
Keys	One key is shared between two or more entities.	One entity has a public key and the other entity has a private key.
Key exchange	Out-of-band.	Symmetric key is encrypted and sent with message; thus, the key is distributed by inbound means.
Speed	Algorithm is less complex and faster.	Algorithm is more complex and slower.
Key length	Fixed-key length.	Variable-key length.
Use	Bulk encryption, which means encrypting files and communication paths.	Key encryption and distributing keys.
Security service provided	Confidentiality and integrity.	Confidentiality, integrity, authentication, and nonrepudiation.

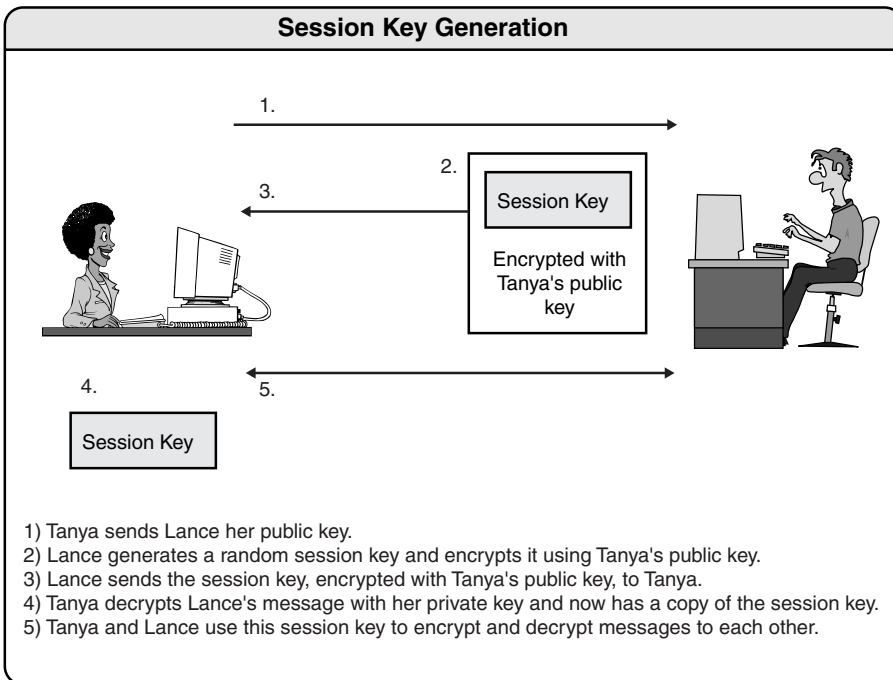


Figure 8-19 A session key is generated so that all messages can be encrypted during one particular session between users.

A session key provides security because it is only valid for one session between two computers. If an attacker captured the session key, she would have a very small window of time to use it to try and decrypt messages being passed back and forth.

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, the resources are released, and the session key is destroyed.

So there are keys used to encrypt data and different types of keys used to encrypt keys. These keys must be kept separate from each other and neither should try to perform the other key's job. A key that has a purpose of encrypting keys should not be used to encrypt data and anything encrypted with a key used to encrypt other keys should not appear in clear text. This will reduce the vulnerability to certain brute force attacks.

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 as a security professional, it is important to understand the difference between the key types and the issues that surround them.

Public Key Infrastructure (PKI)

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. PKI is an ISO authentication framework that uses public key cryptography and the X.509 standard protocols. The framework was set up to enable authentication to happen across different networks and the Internet. Specific protocols and algorithms are not specified, and that is why it 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 was discussed in earlier sections.

There is a difference between public key cryptography and PKI. So to be clear, public key cryptography entails the algorithms, keys, and technology required to encrypt and decrypt messages. PKI is what its name states—it is an infrastructure. The infrastructure of this technology assumes that the receiver's identity can be positively ensured through certificates and that the Diffie-Hellman exchange protocol (or another type of key exchange protocol) will automatically negotiate the process of key exchange. So the infrastructure 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.

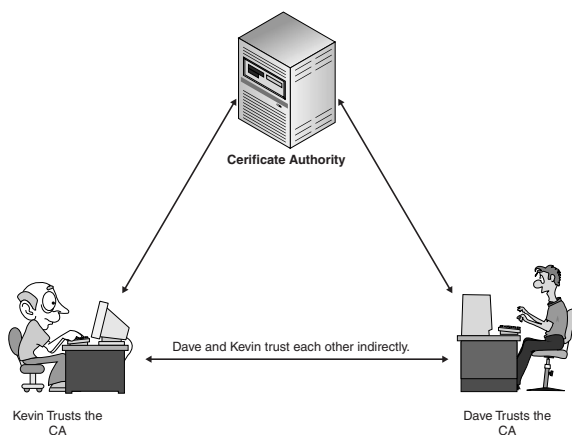
Public key cryptography is one piece in PKI, but there are many other pieces that are required to make up this infrastructure. An analogy is the e-mail protocol Simple Mail Transfer Protocol (SMTP). SMTP is the technology used to get e-mail messages from here to there, but many other things must be in 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 of that individual along with other identifying information. The certificate is signed (digital signature) by a trusted third party or a *certificate authority (CA)*. The CA is responsible for verifying the identity of the key owner. 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 public key. 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 trusted the same CA, then Kevin could retrieve David's public key from that CA and start the process.

Certificate Authorities

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

A CA is an organization that maintains and issues public key certificates. When a person requests a certificate, the CA verifies that individual's identity, constructs the certificate, signs it, delivers 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 message from Kevin, which contains Kevin's public key, David will go back to the CA and basically say, "Hey, is this guy really Kevin Chaisson?" The CA will look up in its database and reply, "Yep, that's him, and his certificate is valid." Then David feels more comfortable and allows Kevin to communicate with him.



Public key cryptography is based on the users trusting the CA, which lets them trust each other indirectly.

The CA can be internal to an organization. This type of setup would enable the company to control the CA server, configure how authentication will take 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 this type of functionality. Some well-known CAs are Entrust and Verisign. Many browsers have several well-known CAs configured by default so the user does not need to figure out how to contact the CA and go through the processes of verifying other users' certificates. It is all taken care of in the background processing of the Web browser.

The CA is responsible for creating and handing out certificates, maintaining them, and revoking them if necessary. Revocation is handled by the *certificate revocation list (CRL)*. This is a list of every certificate that has been revoked for one reason or another. This list is maintained and updated periodically. A certificate may be revoked because the key holder's private key was compromised, the CA discovered that the certificate was issued to the wrong person, or the lifetime of the certificate had expired. An analogy of 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 verifies that the license has not expired. The same thing happens when a person checks with a CA pertaining to another's certificate. If the certificate became invalid for some reason, the CRL is the mechanism for the CA to let others know this information.

Certificates

One of the most important pieces a PKI is its public key certificate. A *certificate* is the mechanism used to associate a public key with a collection of components sufficient to uniquely authenticate the claimed owner. Each certificate has a unique serial number within the CA, which binds that certificate to its particular owner. The most popular public key certificate is the X.509 v3 certificate. 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 8-20.

Registration Authority

As the number of entities that a CA is responsible for grows, sometimes it is logical to offload some of the work to another component. Many large PKI implementations use a *registration authority (RA)*, which performs the certification registration duties. The

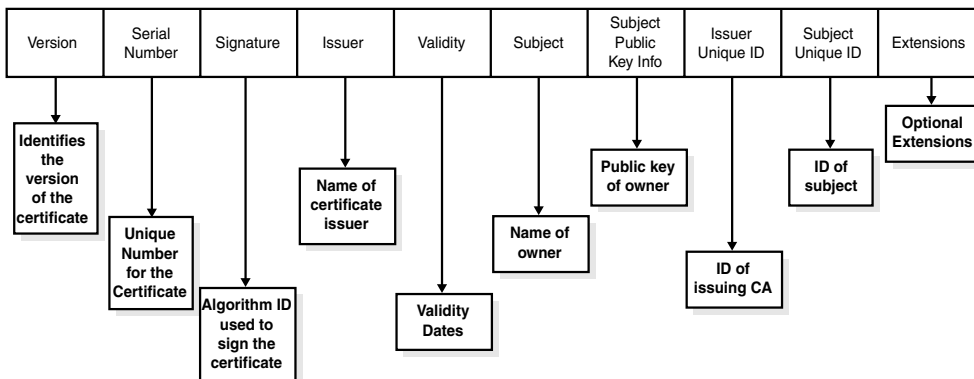


Figure 8-20 Each certificate has a structure with all the necessary identifying information in it.

RA may establish and confirm the identity of an individual, distribute shared keys to end users, initiate the certification process with a CA on behalf of an end user, and perform certificate life cycle management functions. The RA cannot issue certificates, but can act as a middleman between the user and the CA. Sometimes this is beneficial in a distributed environment. If the CA is in New York and there is an office in New Mexico that requires a lot of certificate support, it may be more efficient to have a RA in the New Mexico office, as illustrated in Figure 8-21. When new certificates are needed, users would make requests to the RA and the RA would direct these requests to the CA. This streamlines the communication between the New Mexico office and the CA in New York and lets the RA offload much of the work from the CA.

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.

John needs to establish a public/private key pair for himself, so he makes a request to the CA. The CA requests certain identification from John, like a copy of his driver's license, his phone number, address, and other identification information. Once the CA receives the required information from John and verifies it, the CA registers him in its database and performs a key pair generation. The CA creates a certificate with John's public key and identity information embedded. (The private key is either generated by the CA or on John's machine, which depends on the systems' configurations. If it is created at the CA, it needs to be sent to him by secure means.) Now John is registered and can participate in a PKI. John decides he wants to communicate with Diane so he

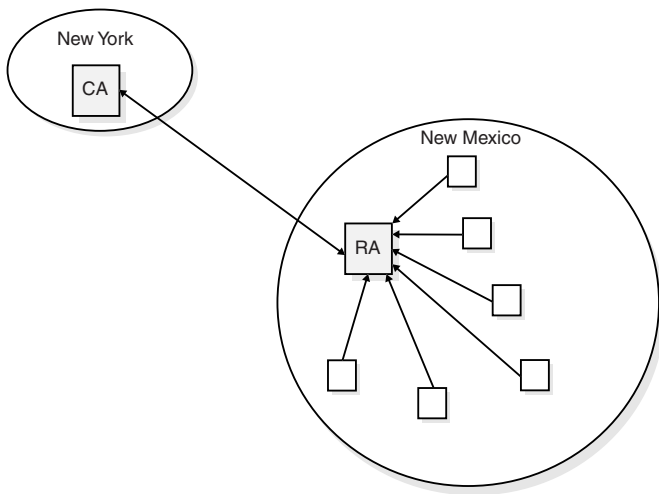


Figure 8-21 An RA can remove much of the load from a CA and reduce network traffic in distributed environments.

requests Diane's public key from the same CA. The CA sends Diane's public key and John uses this to encrypt a session key that will be used to encrypt their messages. John sends the encrypted session key to Diane. John then sends his certificate, containing his public key, to Diane. 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 she makes a request to the CA to see if this certificate is still valid. The CA responds that the certificate is valid so Diane decrypts the session key with her private key. Now they can both communicate using public key cryptography. These concepts are shown in Figure 8-22.

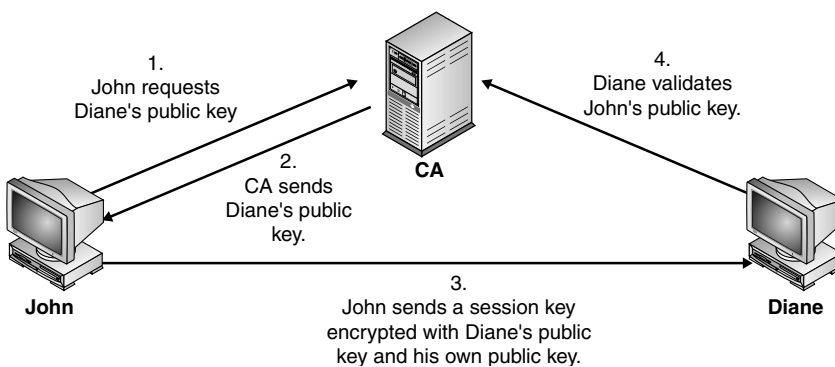


Figure 8-22 CA and user relationships

PKI is 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
- Cross-certification with other CAs
- Timestamping
- Client-side software

PKI supplies the following security services:

- Confidentiality
- Access control
- Integrity
- Authentication
- Nonrepudiation



NOTE Separate Keys: Most implementations of public key cryptography have separate keys for digital signatures and encryption. This separation can add layers of necessary protection. Each key type can have its own expiration date, backup procedures, storage (hard drive, database, or smart card), and strength (64-bit encryption and 128-bit signature). One person may have different digital signatures with different strengths also. Joe may have a digital signature key he uses as an information warfare engineer, another as the lead of logistics, and another as the part owner of a small business. This separation provides more flexibility and the right level of security where it is needed.

References

www.rsa.com/rsalabs/faq/4-1-3-1.html

www.pki-page.org/

www.webtools.com/story/security/TLS20010222S0001

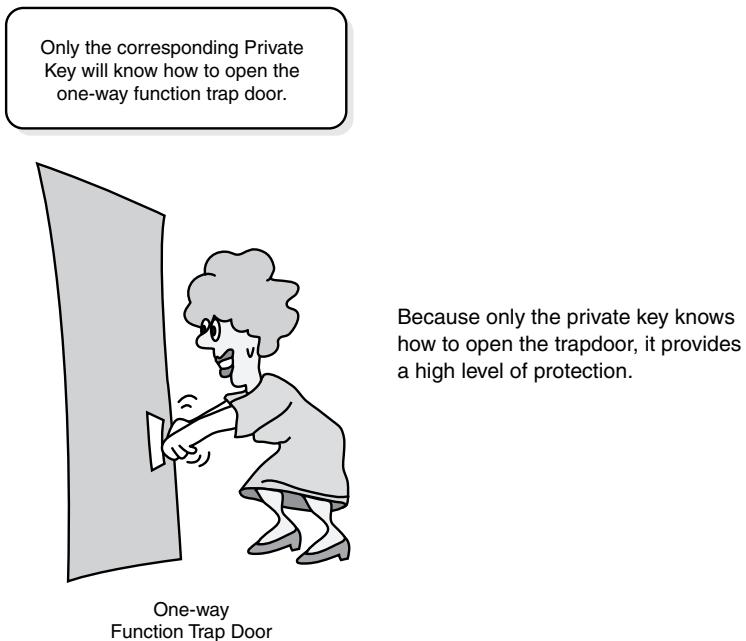
www.nwfusion.com/research/crypto.html

One-Way Function

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 back all the pieces to have the glass again is next to impossible. This concept is similar to how a one-way function is used in cryptography.

The easy direction of computation in a one-way function is like multiplying two large prime numbers. It is easy to multiply the two numbers and get the resulting product, but it is much harder to factor the product and recover the two initial large prime numbers. Many public key encryption algorithms are based on the difficulty of factoring large numbers that are the product of two large prime numbers. So when there are attacks on these types of cryptosystems, the attack is not necessarily trying every possible key value, but trying to factor the large number. So the easy function in a one-way function is multiplying two large prime numbers and the hard function is working backwards by figuring out the large prime numbers that were used to calculate the obtained product number.

Public key cryptography is based on *trapdoor one-way functions*. When a user encrypts a message with a public key, this message is encoded with a one-way function (breaks 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 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 makes a private key a private key.



The crux of this section is that public key cryptography provides security by using mathematical equations that are easy to perform one way (using the public key) and next to impossible to perform the other way (using the private key). An attacker would have to go through a lot of work to perform the mathematical equations in reverse (or figure out the private key).

Message Integrity

Cryptography can detect if a message has been modified in an unauthorized manner in a couple of different ways. The first way is that the message will usually not decrypt properly if parts of it have been changed. The same type of issue happens in compression. If a file is compressed and then some of the bits are modified, either intentionally or accidentally, many times the file cannot be uncompressed because it cannot be successfully transformed from one form to another.

Parity bits have been used in different protocols to detect modifications of streams of bits as they are passed from one computer to another, but parity bits can usually only detect unintentional modifications. Unintentional modifications can happen if there is

a spike in the power supply, if there is interference or attenuation on a wire, or if some other type of physical condition occurs that causes the corruption of bits as they travel from one destination to another. Parity bits cannot identify if a message was captured by an intruder, altered, and then sent on to the intended destination because 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, cryptography is required to successfully detect intentional and unintentional unauthorized modifications to data.

One-Way Hash

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

A *one-way hash* is a function (usually mathematical) that takes a variable-length string, a message, and compresses and transforms it into a fixed-length value referred to as a hash value. A hash value is also called a *message digest*. I know, more confusing names!

The reason to go through these steps is to create a fingerprint of this message. Just as fingerprints can be used to identify individuals, hash values can be used to identify a specific message. If Kevin wants to send a message to Maureen and he wants to ensure that 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 compares her result with the hash value that was sent with the message. If the two values are the same, Maureen can be sure that the message was not altered during transmission. If the two values are different, Maureen knows that the message was altered, either intentionally or unintentionally, and she discards the message.

The hashing function, usually an algorithm, is not a secret—it is publicly known. The secrecy of the one-way hashing function is its “one-wayness.” The function is only run in one direction, not the other direction. This is different than the one-way function used in public key cryptography. In public key cryptography, the security is provided because it is very hard, without knowing the key, to perform the one-way function backwards 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. (Several hashing algorithms are described in the section “Different Hashing Algorithms.”)

The hashing one-way function takes place without the use of any keys. This means that anyone who receives the message can run the hash value and verify the message’s integrity. However, if a sender only wants a specific person to be able to view the hash

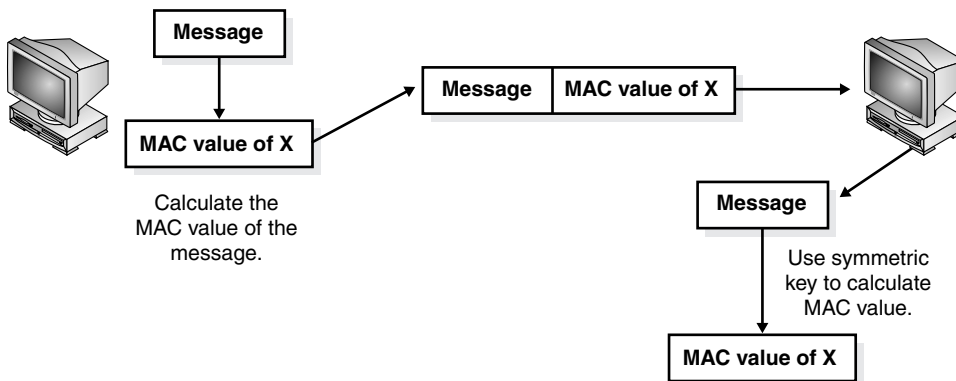


Figure 8-23 The receiver compares her calculated MAC value with the value that was sent with the message.

value sent with the message, the value would be encrypted with a key. This is referred to as the *message authentication code (MAC)*. MAC is the same thing as a one-way hashing function, except that the resulting hash value is the function of the message and the key, as shown in Figure 8-23. This ensures that only the person with the necessary key can verify the integrity of this message.



NOTE A MAC is a key dependent one-way hash function. It has the same functionality as a one-way hash, but it requires a symmetric key to be used in the process and a one-way hash does not. Basically, the MAC is a one-way hash value that is encrypted with a symmetric key.

One-Way Function Used in Encryption versus One-Way Hashing

One-Way Function Used in Public Key Cryptography

- It helps the encryption algorithm to provide confidentiality and authentication, because only the private key can reverse the one-way function to result in plaintext
- The function encrypts in one direction and then decrypts in the reverse direction.

One-Way Hashing Function

- It is never performed in reverse.
- It provides integrity of a message, not confidentiality or authentication.
- The results of a one-way hash is a hashing value.
- It is used in hashing to create a fingerprint for a message.

Digital Signatures

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

A *digital signature* is an encrypted hash value. From our previous example, if Kevin wanted to ensure that the message he sent to Maureen was not modified and he wants her to be sure that 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 with Kevin's public key. She then compares the two values and if they are the same, she can be sure that the message was not altered during transmission. She is also sure that the message came from Kevin because the value was encrypted with his private key.

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 that the value was encrypted with a private key. The steps of a digital signature are outlined in Figure 8-24.

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

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

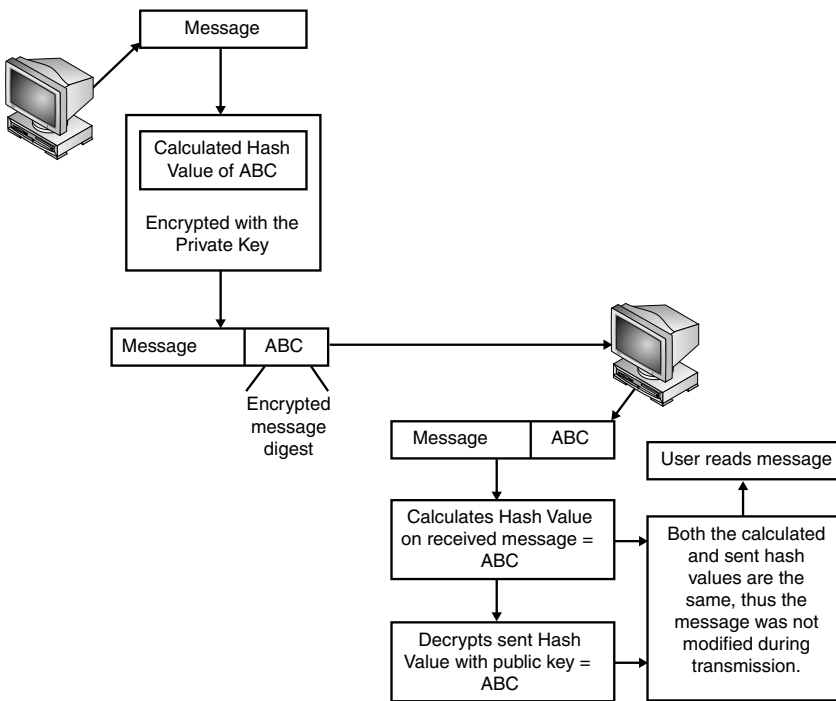


Figure 8-24 Steps of digital signature

It is important to understand that all encryption algorithms cannot 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 8-2 shows the different services provided by the different algorithms.

Table 8-2 Various Functions of Different Algorithms

Algorithm Types	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

(continued)

Table 8-2 Different Functions of Different Algorithms (*continued*)

Algorithm Types	Encryption	Digital Signature	Hashing Function	Key Distribution
Symmetric Key Algorithms				
DES	x			
3DES	x			
Blowfish	x			
IDEA	x			
RC4	x			
SAFER	x			
Hashing Algorithms				
RSA message digest used within RSA operations			x	
Ronald Rivest family of hashing functions MD2, MD4, and MD5			x	
Secure Hash Algorithm (SHA) used with Digital Signature Algorithm [DSA]		x	x	
HAVAL (variable-length hash values using a one-way function design)			x	

Digital Signature

A digital signature is the encrypted hash value of a message. The act of signing means encrypting the message's hash value with a private key, as shown in Figure 8-25.

Digital Signature Standard (DSS)

Because digital signatures can hold such importance on proving who sent what messages and when, the government decided to erect standards pertaining to its functions

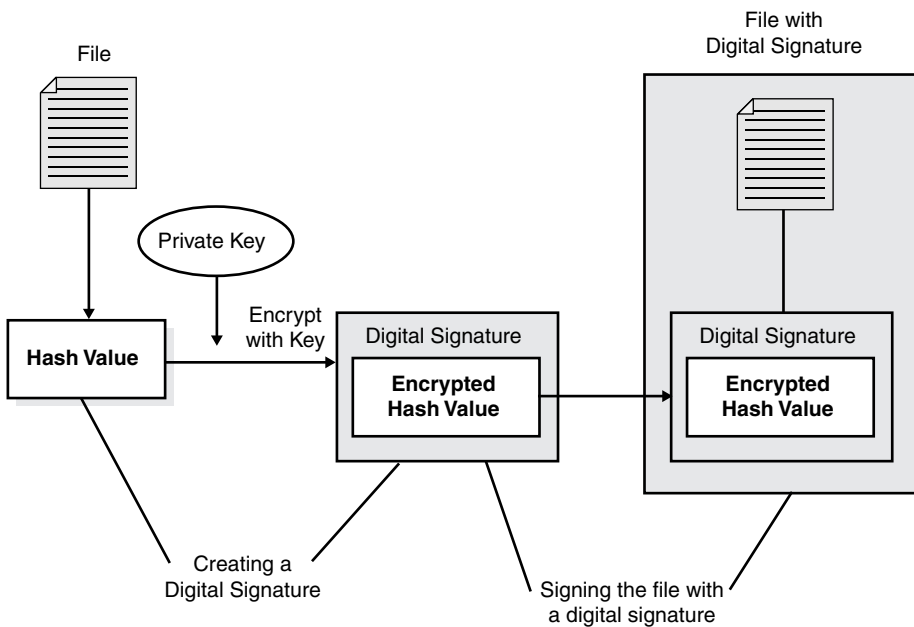


Figure 8-25 Creating a digital signature for a message

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 the Digital Signature Algorithm (DSA) and the Secure Hash Algorithm (SHA). The SHA creates a 160-bit output, which is then inputted into the DSA. The SHA is used to ensure the integrity of the message and the DSA is 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. Unlike RSA, DSA can only be used for digital signatures and is part of the DSS. RSA can be used for digital signatures and message encryption.

Different Hashing Algorithms

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

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

Good cryptographic hash functions should have the following characteristics:

- The hash should be computed on the entire message.
- The hash should be a one-way function so that messages are not disclosed by their signatures.
- It should be impossible, given a message and its hash value, to compute another message with the same hash value.
- It should be resistant to birthday attacks, meaning an attacker should not be able to find two messages with the same hash value.

Table 8-3 and the following sections quickly describe some of the available hashing algorithms used in cryptography today.

MD4

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

Table 8-3 The Different Hashing Algorithms Available

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. Processes text in 512-bit blocks.
HAVAL	One-way function. Variable-length hash value. Modification of MD5 algorithm and provides more protection against attacks that affect MD5. Processes text in 1,024-bit blocks.
SHA	One-way function. Produces a 160-bit hash value. Used with DSA.
SHA-1	Updated version of SHA.

MD5

MD5 is the newer version of *MD4*. It still produces a 128-bit hash, but the algorithm is a bit more complex to make it harder to break than *MD4*. The *MD5* added a fourth round of operations to be performed during the hashing functions and makes several of its mathematical operations carry more steps or more complexity to provide a higher level of security.

MD2

MD2 is also a 128-bit one-way hash function designed by Ron Rivest. It is not necessarily any weaker than the previously mentioned hash functions, but it is much slower.

SHA

SHA was designed by NIST and NSA to be used with the *DSS*. The *SHA* was designed to be used in digital signatures and developed when a more secure digital signature algorithm was required for federal applications.

SHA produces a 160-bit hash value, or message digest. This is then inputted into the *DSA*, which computes the signature for a message. The message digest is signed instead of the whole message because it is a much quicker process. The sender computes a 160-bit hash value, encrypts it with his private key (signs it), appends it to the message, and sends it. The receiver decrypts the value with the sender's public key, runs the same hashing function, and compares the two values. If the values are the same, the receiver can be sure that the message has not been tampered with while in transit.

SHA is similar to *MD4*. It has some extra mathematical functions and produces a 160-bit hash instead of 128-bit, which makes it more resistant to brute force attacks, including birthday attacks. (Birthday attacks are described in "Attacks Against One-Way Hash Functions" section.)

HAVAL

HAVAL is a variable-length one-way hash function and is the modification of *MD5*. It processes message blocks twice the size of those used in *MD5*; thus, it processes blocks of 1,024 bits.

References

- <http://csrc.nist.gov/encryption/aes/>
- www.counterpane.com/tutorials.html
- www.the-search-directory.com/cryptography/
- <http://theory.lcs.mit.edu/~rivest/rfc1321.txt>

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 referred to as a *collision*. If an attacker finds an instance of a collision, he has more information to use when trying to break the cryptographic methods used.

A complex way of attacking a one-way hash function is called the *birthday attack*. Now hold on to your hat while we go through this—it is a bit tricky.

In standard statistics, a birthday paradox exists. It goes something like this:

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, which 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 you finding another person sharing your birthday—thus, the birthday paradox.

So why do we care? Well, it can apply to cryptography also. The main way that 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 exact 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. Then to take this one step further, finding two messages that hash to the same value would only require $2^{n/2}$.

This means that if an attacker has one hash value and wants to find a message that hashes to that same hash value, this process could take him years. However, if he just wants to find any two messages with the same hashing value, it could take him only a couple of hours.

The hash function used in digital signatures usually uses the value of n that is large enough to make it collision free. This would make $2^{n/2}$ practically impossible to guess

or obtain. So the MD5 algorithm that has a 128-bit output will require 2^{64} computations to break. An algorithm that has 160-bit output, like SHA1, will require 2^{80} computations to break. This means that there is less than 1 in 2^{80} chance that someone will break an encryption system. The main point of this paradox and this section is to show how important longer hashing values truly are. A hashing algorithm that has a larger bit output is stronger and less vulnerable to brute force attacks like a birthday attack.

References

www.stack.nl/~galactus/remailers/attack-3.html

www.rsa.com/rsalabs/faq/2-4-6.html

One-Time Pad

A *one-time pad* is a perfect encryption scheme because it is unbreakable and each pad is used exactly once.

A one-time pad uses a truly nonrepeating set of random bits that are combined bit-wise XOR with the message to produce ciphertext. The random key is the same size as the message and is only used once. Because the entire key is random and as long as the message, it is said to be unbreakable even with infinite resources. Each bit in the key is XORed with a bit in the message and this ensures that each bit is encrypted by a nonrepeating pattern of bits. The sender encrypts the message and then destroys the one-time pad and after the receiver decrypts the message, he destroys his copy of the one-time pad.

One-time pads are integrated in some applications. There is a pseudorandom sequence generator that feeds values to the algorithm, which in turn creates the one-time pad and then XORs it to the message. A one-time pad is unbreakable if the same pad is never used more than once and the bits used in the key are truly random. This ensures that even if an attacker intercepted a message, he would not be able to decrypt it because he would have to have the one-time pad value. If an attacker was actually successful in intercepting a copy of the one-time pad key, it would not be useful because the pad is only good for a one-time use.

Let's walk through this. Two copies of a pad containing a set of completely random numbers are created. This set contains at least as many numbers as characters in the message that is to be encrypted. The numbers within the set are produced by a secure random number generator, which can be seeded by the date, time, or other sources like

radioactive decay. The seed is the starting value, which determines all subsequent values in the sequence used to generate the one-time pad.

Although this approach to encryption can provide a very high degree of security, it is impractical in most situations because it is difficult to distribute the pads of random numbers to all the necessary parties. Each possible pair of entities that might want to communicate in this fashion must receive a key that is as long, or longer, than the actual message. This type of key management can be overwhelming and require more overhead than it is worth. The distribution of the pad, or key, can be challenging and the sender and receiver must be perfectly synchronized so that each is using the same pads. The steps of encryption using a one-time pad are shown in Figure 8-26.

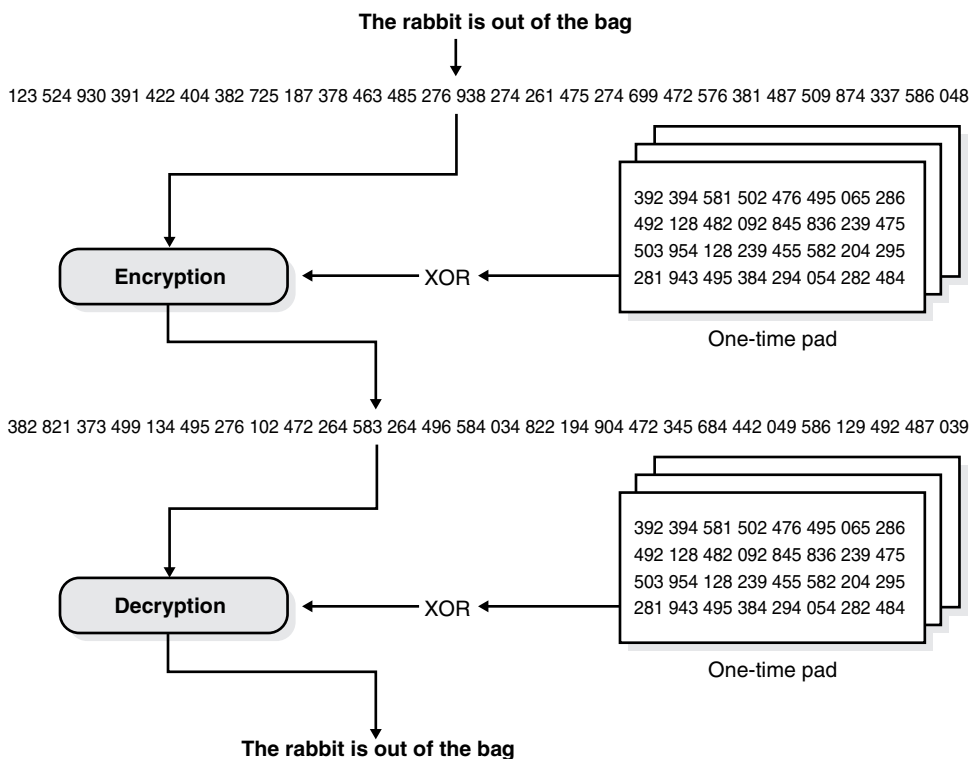


Figure 8-26 A one-time pad

Key Management

I am the manager of all keys! Answer: I am sorry.



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 trust each other to protect their own keys, they trust the administrator that is maintaining the keys, or they 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 have to be distributed to the right entities and updated continuously. The keys need to be protected as they are being transmitted and while they are being stored on each workstation and server. The keys need to 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 is not going to just hang out on the desktop; it needs a secure place within the file system to be stored and used in a control 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 was able to obtain these components, she could masquerade as another user, and decrypt, read, and reencrypt messages that were not intended for her.

Historically, 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 take place by a protocol through automated means and not manually by an individual. The overhead of key management, required security level, and cost-benefit

issues need to be evaluated for a company 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 keys. This method provides an automated method of key distribution. The computer that wants to access a service on another computer will request access via the KDC. The KDC then calculates 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 KDC gets compromised in any way, then all the computers and their services are affected and possibly compromised.

Other key exchange protocols are RSA and Diffie-Hellman, discussed earlier, and a variation of the Diffie-Hellman algorithm called the key exchange algorithm (KEA).

Unfortunately, many companies use cryptographic keys, but rarely change them out, if at all. This is because of the hassle of key management and 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 can have 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 their cryptography keys every six months, whereas an information warfare military unit might change them every day. The important thing is that the keys are being changed in a secure method.

Key management is the most challenging part of cryptography. 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. Keeping keys secret is a challenging task.

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, then 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 if the process depends 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 is encrypted and then the user accidentally loses the necessary key to decrypt it, this information would be lost forever if there was 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. There are different scenarios that can show the need for key recovery or backup copies of keys. If Bob has possession of all the critical bid calculation, stock value information, and the 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. If an employee left the company and encrypted important documents on her computer before departing, the company would probably want to have a way to still have access to that data. Similarly, if the vice president did not know that running a large magnet over the diskette 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 can increase the chance of disclosure, so a company needs to decide if it will have key backups and what precautions will be 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 needs to be recovered, more than one person is required to be involved with this process. The key recovery process could require two other individuals to present their private keys or three individuals to supply their individual PINs. These individuals should not all be members of the IT department. There should be a member from management, maybe an individual from auditing, 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. This is an example of key escrow, which was previously explained in the "Key Escrow" section.

Rules for Keys and Key Management

- 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 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 lifetimes comes to an end.

References

<http://home.ecn.ab.ca/~jsavard/sicrypt.htm>

www.rsa.com/rsalabs/faq/

www.faqs.org/faqs/cryptography-faq/part05/

www.cs.georgetown.edu/~denning/crypto/

www.ssh.fi/tech/crypto/intro.html

Link versus End-to-End Encryption

There are two communication levels at which encryption can be performed, each with different types of protection and implications. These two general modes of encryption implementation are link and end-to-end. *Link encryption* encrypts all the data along a specific communication path like 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. This provides extra protection against packet sniffers and eavesdroppers. In *end-to-end encryption*, the headers, addresses, routing, and trailer information are not encrypted; therefore, attackers can 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 have to be decrypted at each hop so the computer or router knows where to send the packet next. The computer, or router, must decrypt the packet, read the routing and address information within the header, and then reencrypt 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 computers in between the origin and destination just read the necessary routing information and pass the packets on their way. Although link encryption provides extra pro-

tection as it travels through the communication path, it does expose the packets at each computer that has to decrypt it. There is always a little bad with the good, isn't there?

End-to-end encryption is usually initiated at the application layer of the originating computer. It provides more flexibility for the user to be able to determine if certain messages will get encrypted or not. It is called end-to-end because the message stays encrypted from one end of its journey to the other. Which is different than link encryption, which has to decrypt the packets at every computer between the two ends.

Encryption can happen at the highest levels of the OSI model or the lowest levels. If the encryption happens at the lower layers, then it is link encryption and at the higher levels, it is considered end-to-end encryption.

Link encryption is at the physical layer, as depicted in Figure 8-27. Hardware encryption devices interface with the physical layer, which encrypt all data that pass through them. All data, routing, and protocol information are encrypted through these devices if link encryption is in place. Because no part of the data is available to an attacker, she cannot learn basic information about how data flows through the environment. This is referred to as traffic-flow security.

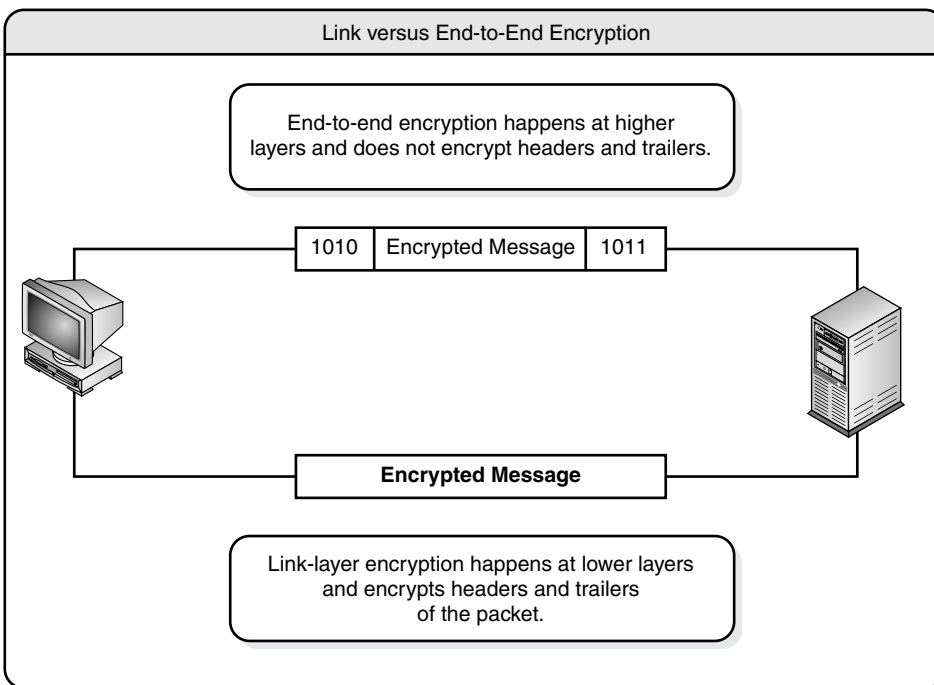


Figure 8-27 Link and end-to-end encryption happen at different OSI layers.



NOTE A hop is a computer that helps a packet get to 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.

Encryption can take place within software or through specialized devices. If a device is going to encrypt the data, then the computer sends its data to the specialized hardware device for encryption before sending it to the lower layers of communication to prepare for transmission.

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

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

- It protects information from start to finish throughout the network.
- It provides more flexibility to the user in choosing what gets encrypted and how.
- Higher granularity of encryption is available because each application or user can use a different key.
- 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 is not encrypted, and therefore not protected.
- The destination system needs to have the same encryption mechanisms to properly decrypt the message.

Advantages of link encryption include the following:

- All data is 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 is more complex because each hop computer must receive a key and when the keys change each must be updated.
- Messages are decrypted at each hop; thus, there are more points of vulnerability.

Reference

www.rand.org/publications/RM/RM3765/RM3765.chapter3.html

Hardware versus 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 when 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.

E-mail Standards

Just like in other types of technologies, cryptography has industry standards and de facto standards. Standards are necessary because they help ensure interoperability between 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 will still need to decide on what type of standard to follow and what type of technology to implement.

The goals of the technology need to be evaluated and a cost-benefit analysis needs to be performed on the competing standards and products within the chosen standards. For cryptography implementation, the company would need to decide on what needs to be protected by encryption, if 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 might service them best. Each type of technology and standard should be understood to help make the most informative decision and each competing product within the chosen technology should be researched and tested before making the final purchase. Cryptography can be a complicated subject, and so is

implementing and maintaining it. Doing homework versus buying into buzzwords and flashy products might help a company reduce its headaches down the road.

The following sections quickly describe some of the most used e-mail standards in use.

Privacy-Enhanced Mail

Privacy-Enhanced Mail (PEM) is an Internet standard to provide secure e-mail over the Internet. 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 procedures developed by several governing groups. PEM can use DES for encryption and RSA for sender authentication and key management. It also provides support for nonrepudiation. The following outlines specific components that can be used in PEM:

- Messages encrypted with DES in CBC mode.
- Authentication provided by MD2 or MD5.
- Public key management provided using RSA.
- X.509 standard used for certification structure and format.

Message Security Protocol

The *Message Security Protocol (MSP)* is the military's PEM. It was developed by the NSA and is an X.400-compatible application level protocol used to secure e-mail messages. MSP can 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

www.cs.auckland.ac.nz/~pgut001/tutorial/

www.informweb.com/webportal/articles/tosecs.htm

Pretty Good Privacy (PGP)

Pretty Good Privacy (PGP) was designed by Phil Zimmerman as a freeware e-mail security program and released in 1991. It was the first widespread public key encryption program. PGP is a complete working system that uses cryptographic protection to pro-

protect e-mail and files. It mainly uses RSA public key encryption for key management and IDEA symmetric cipher for bulk encryption of data, although the user has the option of picking different types of algorithms to use. PGP can provide confidentiality through the IDEA encryption algorithm, integrity by using the MD5 hashing algorithm, authentication by using the public key certificates, and nonrepudiation through the use of cryptographically signed messages. (PGP enables different algorithms to be plugged in, so some implementations may use different algorithms than are listed here.)

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, but 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 than the CA approach where no one trusts each other; they only trust the CA.

For example, if Mark and Mike want to communicate using PGP, Mark can give his public key to Mike. Mike signs Mark's key and keeps a copy for himself. Then Mike gives a copy of his public key to Mark so they can start communicating securely. Later, Mark would like to communicate with Joe, but Joe does not know Mark, and does not know if he can trust him. Mark sends Joe his public key, which has been signed by Mike. Joe has Mike's public key, because they have communicated before, and trusts Mike. Because Mike signed Mark's public key, Joe now trusts Mark also and sends his public key and begins communicating with him.

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

Each user keeps a collection of signed public keys he has received from other users in a file referred to as a *key ring*. 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 Tom, whom he does not trust much at all. There is also a field indicating who can sign other keys within in 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 who he trusts to sign other people's keys. If the field indicates that Steve trusts Liz enough to sign another person's key, then Steve will accept Kevin's key and communicate with him.

However, if Steve receives a key from Kevin and it is signed by untrustworthy Tom, then 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, like a CA, certain standardized functionality is harder to accomplish. If Steve lost his private key, it means anyone else trusting his public key must be notified 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 will be told not to trust it when the other users contacted the CA. In the PGP world, this is not as centralized and organized. Steve can send out a key revocation certificate, but there is no guarantee that 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 standard on the Internet.

References

<http://web.mit.edu/network/pgp.html>

www.pgpi.org/doc/pgpintro/

<http://axion.physics.ubc.ca/crypt.html#PGP>

www.pgpi.org/

Internet Security

The Web is not the Internet. The Web runs on top of the Internet, in a sense. The Web is the collection of Hypertext Transfer Protocol (HTTP) servers that hold and process Web sites that 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 touch upon in Chapter 2.) The Web sites look the way they look because the creator used a language that dictates the look, feel, and functionality of the page. The Web browser lets users 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 lot of different protocols and have the capability to process many types of commands, but they do not understand them all. For the protocols or commands they do not know how to process, the user must download a viewer or plug-in. This is a quick and easy way to expand the functionality of the browser by

installing a modular component of code that integrates itself into the system or browser. This has caused serious security compromises because the payload of the module can easily carry viruses and malware that the users do not know about until it is already installed and has started its damage.

Start with the Basics

Why do we even connect to the Internet? This is a basic question at first, but as we dive deeper into the question, the complexity creeps in. We connect to download MP3s, check e-mail, order security books, look at Web sites, communicate with friends, and much more. But what are we really doing? We are using services provided by a computer's protocols and software. The services can be file transferring provided by FTP, remote connectivity provided by Telnet, Internet connectivity provided by HTTP, secure connections provided by SSL, and much, much more. Without these tools, 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 needs to implement these decisions by controlling services that can be used inside and outside the network. There are different ways of restricting services: allow certain services to only run on a particular system and restrict access to that system, employ a secure version of a service, filter the use of services, or block them altogether. These decisions will determine how secure the site will be and indicate what type of technology is needed to provide this type of protection.

HTTP

TCP/IP is the protocol of the Internet and HTTP is the protocol of the Web. HTTP sits on top of TCP/IP. When a user clicks her mouse on a link within a Web page, 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 maintaining the connection between the user and the server and the IP protocol makes sure that it 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 that the origin and destination are correct and that no packets are lost along the way, and upon arrival of the destination, HTTP presents the payload, which can be 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 site, that Web server finds the requested Web site, presents it to the user, and then breaks the connection. If the user requests a link within the newly received Web page, a new connection

has to be set up, the request goes to the Web server, and the Web server sends the requested item and breaks the connection.

S-HTTP

Secure Hypertext Transport Protocol (S-HTTP) is HTTP with added on security features. It was developed to provide secure communication between a client and a server over the Internet. The client and server both have a list of cryptographic preferences and keying material. When a client makes a request to a server and the server deems that this type of communication should be protected, the server will query the client about the type of encryption methods it is configured to use. Once the client and server agree upon a specific encryption method, the client sends the server its public key. The server generates a session key from this public key, encrypts the session key with the client's public, and sends it back. From here on out, the client and server encrypt their messages with the newly calculated session key.

S-HTTP can also provide data integrity and sender authentication capabilities. S-HTTP computes a hash value of the message and the value can then be digitally signed. It was stated earlier that HTTP is a stateless protocol, meaning that after an operation is complete, the connection is disconnected. This is not the case with S-HTTP because it would require too much overhead if the client and server had to handshake and agree upon security parameters for each and every operation.

S-HTTP can support multiple encryption modes and types. It can use public key technology, PEM, and even symmetric key encryption. S-HTTP does not require the client to obtain a public key certificate if symmetric session key operations are allowed. This is a much less secure way of communicating, but it shows the flexibility of S-HTTP.

HTTPS

There is a difference between S-HTTP and HTTPS. S-HTTP is a technology that protects each message that is sent between two computers. *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 needs to be encrypted, then HTTPS is used, which is SSL over HTTP.

SSL

Secure Sockets Layer (SSL) is similar to S-HTTP, but it protects a communication channel instead of individual messages. It uses public key encryption and provides data encryption, server authentication, message integrity, and optional client authentication. When

a client accesses a Web site, it is possible for that Web site to have 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 that a secure session needs to be established, and the client sends its public key and 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 server creates a session key and encrypts it with its private key. The client decrypts the session key with the server's public key and the two use this session key throughout the session.

Just like S-HTTP, SSL keeps the communication path open until one of the parties requests to end the session. Usually the client will click on a different URL, and the session is complete.

SSL protocol requires an SSL-enabled server and browser. SSL will provide security for the connection but does not provide security for the data once it is received. This means the data is encrypted while it is being transmitted, but once it is received by a computer, it is no longer encrypted. 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.

In the protocol stack, SSL lies beneath the application layer and above the transport layer. This ensures that SSL is not limited to specific application protocols and can still use the communication transport standards of the Internet.

The user can verify a secure connection by looking at the URL to see that it states `https://`. The same is true for a padlock or key icon, depending on the browser type, at the bottom corner of the browser window.

Reference

<http://csgrad.cs.vt.edu/~mlorch/securityprotocols/6.6.html>

MIME

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 multimedia attachment, MIME dictates how that portion of the message should be handled.

When a user requests a file from a Web server that contains an audio clip, graphic, or some other type of multimedia component, the server 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 our example, the file's name might be `stuff.jpeg`. The user's system will see the extension `jpeg`, or see that 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 the Explorer will open and present the picture to the user.

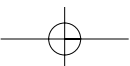
Sometimes systems either do not have an association for a specific file type or do not have the necessary helper program necessary to review and use the contents of the file. When a file has an unassociated icon assigned to it, it might require the user to choose the Open With command and choose an application in the list to associate this file with that program. So when the user double-clicks on 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 wave files.

So 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.

S/MIME

Secure MIME (S/MIME) is a standard for encrypting and digitally signing electronic mail that contains attachments and 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 messages.



References

www.imc.org/smime-pgpmime.html
www.rsa.com/standards/smime/faq.html
<http://nsi.org/Library/Internet/security.htm>
www.ece.umn.edu/users/kjhan/security/

SET

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 the standard for quite sometime. Although SET provides a very 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 developed to send encrypted credit card numbers over the Internet. It is comprised of three main parts: the electric wallet, the software running on the merchant's server at its Web site, and the payment server that is located at the merchant's bank.

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

Let's say Tanya wants to buy her mother a gift from a Web site using her electric wallet. When she finds the perfect gift and decides to purchase it, her encrypted credit card information is sent to the merchant's Web server. The merchant does not decrypt this 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.

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 and they do not feel motivated enough to move to a different and more encompassing technology.

References

www.bankinfo.com/ecommm/setpart1.html

www.sans.org/infosecFAQ/covertchannels/SET.htm

www.cs.jcu.edu.au/~pei/cryptography.htm

Cookies

Hey, I found a Web site that is giving out free cookies! Answer: Great, I will bring the milk!

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

Cookies are text files that a browser maintains on a user's hard drive. There are different uses for cookies, but they are mainly 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 user's specific customization for certain sites. For example, if Emily goes to mainly Christian sites on the Internet, those sites will most likely be recording this information and the types of items in which she shows most interest. Then when Emily comes back to one of the same or similar sites, it will retrieve her cookies, find that she has shown interest in Christian books in the past, and present her with their line of Christian books. This increases the likelihood of Emily purchasing a book of her likening. 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, this data is 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 is extracted and the totals are added.

Cookies can also be used as timestamps to ensure that a session between a user and a server is restricted to a specific length of time. For example, if a user enters her credentials to access her banking information through her online bank account, she will need to be authenticated. Once she is authenticated, the server puts a cookie on her hard drive with a timestamp of four minutes. When she requests to go to another secure page within the site, the server will query the cookie to see if their session has timed out yet. If the session did time out, the user is asked to reenter her credentials to initiate another session.

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 dif-

ferent accounts on the Internet. The cookies that contain sensitive information should be encrypted by the server on the site that distributed them, but this does not always happen and a nosy attacker can 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 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 needs to perform correctly and to be able to provide the user with the services she requested.

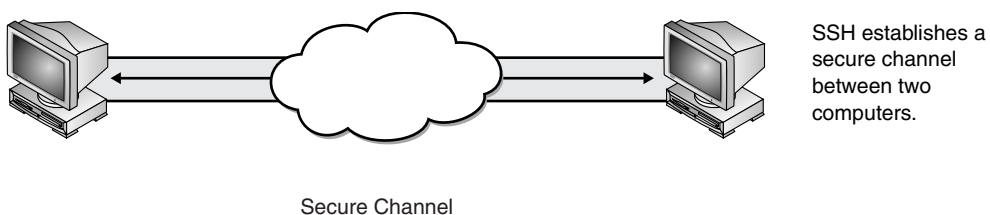
There are third-party products that can limit the type of cookies downloaded, hide the user's identities as he travels from one site to the next, or mask the user's e-mail addresses and the mail servers he uses.

SSH

Secure Shell (SSH) functions as a type of tunneling mechanism that provides terminal-like access to remote computers. SSH is a program that can be used to log into another computer over a network. 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 provides the same type of functionality that SSH provides 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 a session key that will be used during the session to encrypt and protect the data that is exchanged. The steps of an SSH connection are outlined in Figure 8-28.

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



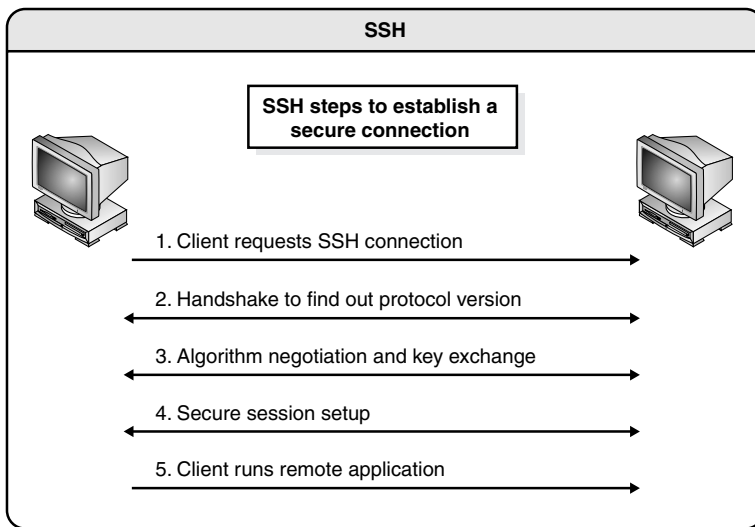


Figure 8-28 SSH is used for remote terminal-like functionality.

References

www.uni-karlsruhe.de/~ig25/ssh-faq/

www.onsight.com/faq/ssh/ssh-faq.html

http://wks.uts.ohio-state.edu/sysadm_course/html/sysadm-558.html

IPSec

The *Internet Protocol security (IPSec)* protocol is 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 secure network layer transport. It is more flexible and less expensive than application and link-layer encryption methods.

IPSec has strong encryption and authentication methods that employ public key cryptography. Although it can be used to enable communication between two computers, it is usually used to establish virtual private networks (VPNs) between networks across the Internet.

IPSec is not a strict protocol that dictates the type of algorithm, keys, and authentication method to be used, but 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 the *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*, where the payload of the message is encrypted, and *tunnel mode*, where the payload and the routing and header information is also encrypted. The transport mode encrypts the actual message information so that it cannot be sniffed and uncovered by an unauthorized entity. The tunnel mode provides a higher level of protection by also protecting the header and trailer data that an attacker may find useful. Figure 8-29 shows the high-level view of the steps of setting up an IPSec connection.

Each device will have one *security association (SA)* for each session that it uses. The SA is critical to the IPSec architecture and is a record of the configurations the device needs to support an IPSec connection. The SA can contain the authentication and encryption keys, the agreed upon algorithms, key lifetime, and the source IP address. When a device receives a packet on the IPSec protocol, it is the SA that tells the device what to do with the packet. So if device B received a packet from device C via IPSec, device B will look to the 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.

A device will have one SA for each connection. It will have one SA for outbound traffic and a different SA for inbound traffic. If it is connecting to three devices, it will have 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? Well, with the mighty *security parameter index (SPI)*, that's how. Each device has an SPI index that keeps track of the different SAs and tells the device which one is appropriate to invoke for the different packets it receives. The relationships between the SPI and the different SAs are depicted in Figure 8-30.

The AH protocol can authenticate the sender of the packet by user or source IP address. The ESP protocol can provide authenticity, integrity, and confidentiality if the devices are configured for this type of functionality. If both protocols are used, the following steps outline the process a receiving device goes through once it receives a packet:

1. Identify the appropriate SPI, SA, secret key, and algorithm (MD5 or SHA-1).
2. Calculate the hash value on packet to authenticate source and verify data integrity.
3. Authenticate the source.
4. Identify the correct cryptographic algorithm (DES or 3DES) and secret key.
5. Decrypt the message.

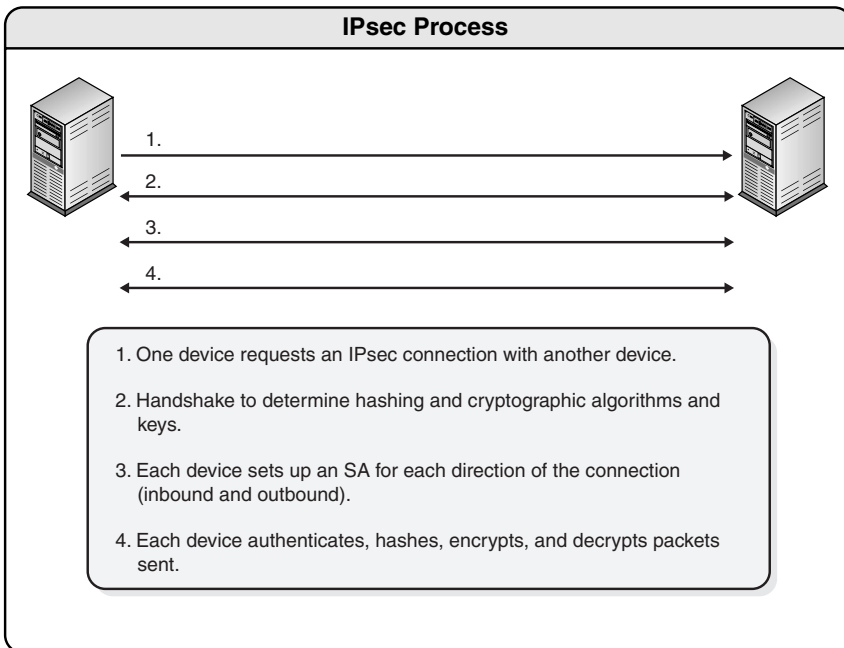


Figure 8-29 Steps that two computers follow when using IPsec

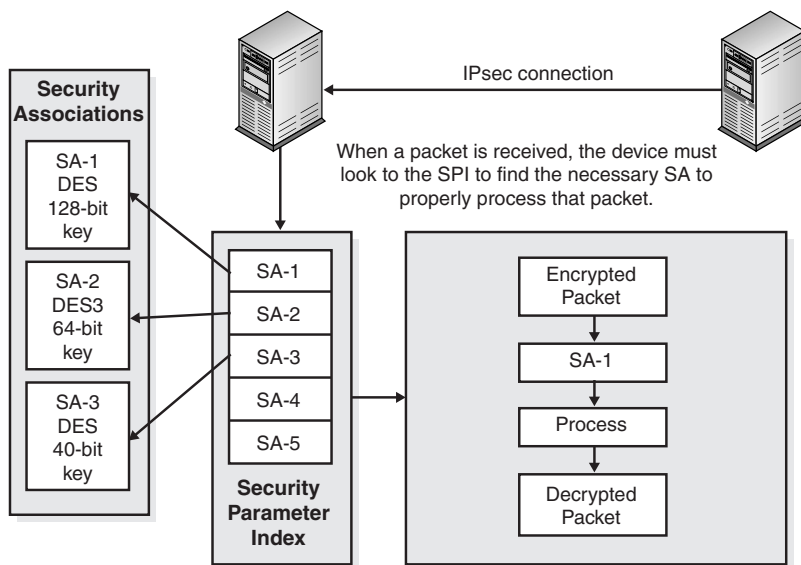


Figure 8-30 The SPI and SA help the system to encrypt and decrypt when using IPsec.

Because IPsec is a framework, it does not dictate what hashing and encryption algorithms are to be used or how keys are to be exchanged between devices. Key management can be handled through manual process or automated by a key management protocol. The *Internet Security Association and Key Management Protocol (ISAKMP)* is an authentication and key exchange architecture that is independent of the type of keying mechanisms used. Basically, the ISAKMP provides the framework and companies can choose how they will deal with key exchanges in their environments and how they will work within the ISAKMP framework.

For more in-depth information please refer to the following references.

References

www.ietf.org/html.charters/ipsec-charter.html

www.cs.arizona.edu/xkernel/www/ipsec/ipsec.html

www.cisco.com/warp/public/cc/so/neso/sqso/eqso/ipsec_wp.htm

www.counterpane.com/ipsec.html

Attacks

Eavesdropping, network sniffing, and capturing data as it passes over a network is considered passive 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 methods are put in place to try and 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 gathering data. Passive attacks are usually used to gain information prior to carrying out an active attack. The following sections go over active attacks that can relate to cryptography.

Ciphertext-Only Attack

In this type of an 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 plaintext of the messages by figuring out the key used in the encryption process. Once the attacker figures out the key, she can now decrypt all other messages encrypted with the same key.

Known-Plaintext Attack

In this type of attack, 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 that other messages can be deciphered and read.

Chosen-Plaintext Attack

In this attack, the attacker has the plaintext and ciphertext, but what makes this type of attack different is that she can choose the plaintext that gets encrypted. This gives the attacker more power and possibly a deeper understanding of the way that the encryption process works so that she can gather more information about the key that is being used. Once the key is discovered, other messages encrypted with that key can be decrypted.

Chosen-Ciphertext Attack

In this attack, 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.

These are the definitions, but how do these attacks actually get carried out? An attacker may make up a message and send it to someone she knows will encrypt the message and then send it out to others. In that case, the attacker knows the plaintext and can then capture the message as it is being transmitted, which gives her the ciphertext.

Also, messages usually start with the same type of beginnings and ends. 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 attack has some of the plaintext (the data that is 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 and trial-and-error attempts. Known-plaintext attacks were used by the United States against the Germans and the Japanese during World War II.

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, the strongest and best engineered algorithms are the ones that are 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 used by the NSA. Because the sensitivity level of what they are encrypting is so important, they want as much of the process as secret as possible. It does not mean that their algorithms are weak because they are not released for public examination and analysis. Their algorithms are developed, reviewed, and tested by many top cryptographic smarty-pants and are of very high quality.

Man-in-the-Middle Attack

If David is eavesdropping on different conversations that happen over a network and would like to know the juicy secrets that Lance and Tanya pass between each other, he can perform a *man-in-the-middle* attack. The following are the steps for this type of attack:

1. Tanya sends her public key to Lance and David intercepts this key and sends Lance his own public key. Lance thinks he has received Tanya's key, but in fact received David's.
2. Lance sends Tanya his public key. David intercepts this key and sends Tanya his own public key.
3. Tanya sends a message to Lance, encrypted in 'Lance's' public key. David intercepts the message and can decrypt it because it is encrypted with his own public key, not Lance's. David decrypts it with his private key, reads the message, and reencrypts it with Lance's real public key and sends it to Lance.
4. Lance answers Tanya by encrypting his message with 'Tanya's' public key. David intercepts it, decrypts it with his private key, reads the message, and encrypts it with Tanya's real public key and sends it on to Tanya.

Many times public keys are kept on a public server for anyone to access. David can intercept queries to the database for individual public keys or David can substitute his public key in the database itself in place of Lance and Tanya's public keys.

The SSL protocol has been known to be vulnerable to some man-in-the-middle attacks. The attacker injects herself right at the beginning of the authentication phase so that she obtains both parties' public keys. This enables her to decrypt and view messages that were not intended for her.

Using digital signatures during the session-key exchange can circumvent the man-in-the-middle attack. If using Kerberos, when Lance and Tanya obtain each other's public keys from the KDC, the public keys are signed by the KDC. Because Tanya and Lance have the public key of the KDC, they both can decrypt and verify the signature on each

other's public key and be sure that it came from the KDC itself. Because David does not have the private key of the KDC, he cannot substitute his public key during this type of transmission.

When Lance and Tanya communicate, they can use digital signatures, which means they sign the message with their private keys. Because David does not have Tanya or Lance's private key, he cannot intercept the messages, read them, and encrypt them again as he would in a successful man-in-the-middle attack.

Dictionary Attacks

If Daniel steals a password file that is filled with one-way function values, how can this be helpful to him? He can take 100,000, or 1,000,000 if he is more motivated, of the most commonly used passwords and run them through the same one-way function and store them in a file. Now Daniel can compare his file results of the hashed values of the common passwords to the password file he stole from an authentication server. The ones that match will correlate to the passwords he entered as commonly used passwords.

For example, Table 8-4 represents Daniel's homemade password file that he ran through a one-way function. Table 8-5 represents the stolen password file.

Comparing the two results, Daniel now knows that STomanio is using the password Computer02, StSheperson is using Server1, and that SFoster is using the password Debbie04. Now Daniel can log on as these individuals and access the network resources using their security credentials.

Table 8-4 Daniel's Homemade Password File

Commonly Used Passwords	One-way Function Value
Password	abc
Kristy01	l24
Cowboys	8yt
Computer02	83d
Administrator	dks
Server1	qwk
Lab003	39c
Elvis03	Dk7
Debbie04	3k5

Table 8-5 The Password File that Daniel Stole

User	One-way Function Value
EmGorenz	329
MitHockabout	aks
MarFairbairn	l2p
STomaino	83d
SKowtko	00d
StSheperson	qwk
DKress	8sn
DFerguson	0d0
SFoster	3k5

This process is an example of how dictionary attacks take place. Most of the time the attacker does not need to come up with thousands or millions of passwords. This work is already done and is readily available at many different hacker sites on the Internet. The attacker only needs a dictionary program and he can then insert the file of common passwords, or dictionary file, into the program and run it against a captured password file.

Replay Attack

A big concern in distributed environments is replay attacks. Kerberos is especially vulnerable to this type of attack. A *replay attack* is when an attacker copies a ticket and breaks the encryption and then tries to impersonate the client and resubmit the ticket at a later time to gain unauthorized access to a resource.

Replay attacks do not only happen in Kerberos. Many authentication protocols are susceptible to these types of attacks because the goal of the attacker is usually to gain access to authentication information of an authorized person so that she can turn around and use it herself to gain access to the network. The Kerberos ticket is a form of authentication credentials, but an attacker can also capture passwords or tokens as they are transmitted over the network. Once captured she will resubmit the credentials, or replay them, in the hopes of being authorized.

Timestamps and sequence numbers are two countermeasures to the replay vulnerability. Packets can contain sequence numbers so each machine will be expecting a specific number on each receiving packet. If a packet has a sequence number that had been



previously used, this is an indication of a replay attack. Packets can also be time-stamped. A threshold can be set on each computer to only accept packets within a certain time frame. If a packet is received that is past this threshold, it can help identify a replay attack.

Summary

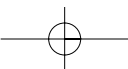
Cryptography has been used in one form or another for over 4,000 years and the attacks on cryptography have probably been in process for 3,999 years and 364 days. As one group of people works to find new ways to hide and transmit secrets, another group of people is right on their heels finding holes in the newly developed ideas and products. This can be viewed as evil and destructive behavior, or hackers can be viewed as the thorn in the side of the computing world that requires them to build better and more secure products and environments.

Cryptographic algorithms provide the underlining tools to most security protocols used in today's infrastructures. The algorithms work off of different mathematical functions and provide different types of functionality and different levels of security. A big leap was made when encryption went from 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 different types of users all over the world.

Encryption can be supplied at different layers of the OSI model by a range of different 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, security professionals who consult 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.
- The process of encryption turns plaintext into ciphertext and decryption transforms ciphertext back into the original plaintext.
- 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 that the sender cannot later falsely deny sending a message and the receiver cannot deny receiving the message.
- Key clustering is an instance when two different keys generate the same ciphertext.
- The range of possible keys is referred to as the keyspace. A larger keyspace and the full use of the keyspace allows more random keys to be created. This brings higher security.
- The two basic types of encryption ciphers are substitution and transposition. Substitution ciphers change a character (or bit) out for another and 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 form, like a graphic, wave file, or document. This method is used to hide the very existence of the data.
- The Clipper Chip was an encryption chip the U.S. government wanted to implement into many American-made devices so that they could listen to communication that contained suspected information about illegal activities.
- The Clipper Chip used the SkipJack algorithm, which was developed by the NSA to enable the government to decrypt any traffic encrypted using the Clipper Chip.
- Key escrow is a practice that splits up the necessary key required to decrypt information. Different agencies, or entities, hold onto the different pieces and come together when decryption is necessary.
- Fair cryptosystems also separate the necessary key required for decryption, but this method takes place in software encryption processes using public key cryptography, whereas key escrow is mainly used when hardware encryption chips are used.
- A key is a random string of bits that is 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 key distribution, scalability, and key secrecy. 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, and RC4.
- Asymmetric algorithms are used to encrypt keys and symmetric algorithms are used to encrypt data.
- If a user encrypts a secret key with his private key, it can only be decrypted by his public key.
- Public and private keys can be used for encryption and decryption processes. If a message is encrypted with a public key, it is decrypted with a private key and vice versa.
- Asymmetric key algorithms are much slower than symmetric key algorithms but can provide confidentiality, authentication, and nonrepudiation services.
- Examples of asymmetric key algorithms include RSA, ECC, Diffie-Hellman, El Gamal, and DSS.
- 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.
- Both stream and block ciphers algorithms are usually 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.
- When DES was 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 block symmetric 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 (ECC) are used as asymmetric algorithms and can provide digital signature, secure key distribution, and encryption functionality. It uses much less resources, which makes it better for wireless device and cell phone encryption use.
- When symmetric and asymmetric key algorithms are used together, this is called a hybrid system or public key cryptography. The asymmetric algorithm encrypts the symmetric secret key and the secret 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 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 to revoke certificates when they have been compromised in some fashion.
- A certificate is the mechanism the CA uses to associate a public key to a person's identity.
- A registration authority (RA) offloads some of the CA's workload by confirming individual identities, distributing shared keys, and submitting requests to the CA on behalf of the users. However, it cannot issue certificates to users.
- A one-way function is a mathematical function that is easier to compute in one direction than in the opposite direction.
- Public key encryption algorithms are based on one-way trapdoor functions. When a message is encrypted with a public key, it is encoded with a one-way function and with a trapdoor. Only the private key knows how to use this trapdoor and decrypt the message.
- Message integrity can be ensured by using hashing algorithms.
- 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.

- When using hash algorithms, the sender runs the message through a hashing algorithm and sends the resulting value with the message to the receiver. The receiver runs the same algorithm and compares the two results. If the results are the same, the receiver can be sure the message was not modified in transit.
- Examples of hashing algorithms include SHA, MD2, MD4, MD5, and HAVAL.
- HAVAL produces a variable-length hash value, whereas the others produce a fixed-length value.
- SHA produces a 160-bit hash value and is used with the Digital Signature Algorithm (DSA).
- A birthday attack is an attack on hashing functions through brute force. The attacker tries to find two messages with the same hashing value.
- A one-time pad uses a pad, or key, 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 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, 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 exchange protocol and does not provide encryption for data.
- Encryption can happen at the physical layer, which is link encryption, or at the application layer, which is end-to-end encryption.
- 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 a freeware e-mail security program that uses public key encryption. It uses a web of trust instead of the hierarchical structure used in public key cryptography.

- S-HTTP provides protection for each message that is sent between two computers, but not the actual link. HTTPS protects the communication channel. HTTPS means HTTP is using 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 transaction over the Internet.

Questions

Please remember that these questions are formatted and asked in a certain way for a reason. The questions and answers may seem odd or vague, but this is what you will see on the actual CISSP test.

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 brute force attacks have increased because _____.
 - a. Of increased use of permutations and transpositions in algorithms.
 - b. As algorithms get stronger, they get less complex, and thus more susceptible to attacks.
 - c. Of the increase in processor speed and power.
 - d. Of the reduction in key length 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 is computationally infeasible to find the corresponding message.
 - c. It is computationally infeasible 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. standard developed for creating secure message digests?
 - a. Data Encryption Standard
 - b. Digital Signature Standard
 - c. Secure Hash Algorithm
 - d. Data Signature Standard
6. If an attacker stole a password file that contained one-way encrypted passwords, what type of attack would she perform to find the encrypted passwords?
 - a. Man-in-the-middle attack
 - b. Birthday attack
 - c. Denial of service attack
 - d. Dictionary attack
7. What is an advantage of RSA over the DSS?
 - a. It can be provide digital signature and encryption functionality.
 - b. It uses fewer resources and encrypts quicker because it uses symmetric keys.
 - c. It is a block cipher versus 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 into different programs.
 - b. It can be used by some countries to be used 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 DES key?
 - a. 56
 - b. 64
 - c. 32
 - d. 16
12. When 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. What is the function of 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 the acronym DEA stand for?
 - a. Data Encoding Standard
 - b. Data Encoding Application
 - c. Data Encryption Algorithm
 - d. Digital Encryption Algorithm
16. Who was involved in developing the first public key encryption system?
 - a. Adi Shamir
 - b. Ross Anderson
 - c. Bruce Schneier
 - d. Martin Hellman
17. What process 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. Time it takes to encrypt and decrypt the same plaintext
 - b. Time it takes to break the encryption
 - c. Time it takes to implement 16 rounds of computation
 - d. Time it takes to apply substitution functions

Answers

- | | | | |
|-------|--------|--------|--------|
| 1. A. | 7. A. | 12. C. | 17. D. |
| 2. C. | 8. C. | 13. B. | 18. A. |
| 3. D. | 9. C. | 14. D. | 19. B. |
| 4. C. | 10. D. | 15. C. | 20. D. |
| 5. B. | 11. A. | 16. D. | 21. B. |
| 6. D. | | | |