CHAPTER 16
Security

Objectives

After studying this chapter, the student should be able to:

- **Define security goals—confidentiality, integrity, and availability.**
- **Show how confidentiality can be achieved using symmetric-key and asymmetric-key cipher.**
- **Discuss other aspects of security: message integrity, message authentication, digital signature, entity authentication, and key management.**
- **Discuss the use of Firewalls to protect a system from harmful message.**

16-1 INTRODUCTION

We are living in the information age. We need to keep information about every aspect of our lives. In other words, information is an asset that has a value like any other asset. As an asset, information needs to be secured from attacks. To be secured, information needs to be hidden from unauthorized access (*confidentiality*), protected from unauthorized change (*integrity*), and available to an authorized entity when it is needed (*availability*).

During the last three decades, computer networks have created a revolution in the use of information. Information is now distributed. Authorized people can send and retrieve information from a distance using computer networks. Although the three above-mentioned requirements changed, they now have some new dimensions. Not only should information be confidential when it is stored; there should also be a way to maintain its confidentiality when it is transmitted from one computer to another.

16.1 Security goals

We will first discuss three security goals: ***confidentiality***, ***integrity***, and **availability.**

Confidentiality

Confidentiality, keeping information secret from unauthorized access, is probably the most common aspect of information security: we need to protect confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information.

Integrity

Information needs to be changed constantly. In a bank, when a customer deposits or withdraws money, the balance of their account needs to be changed. Integrity means that changes should be done only by authorized users and through authorized mechanisms.

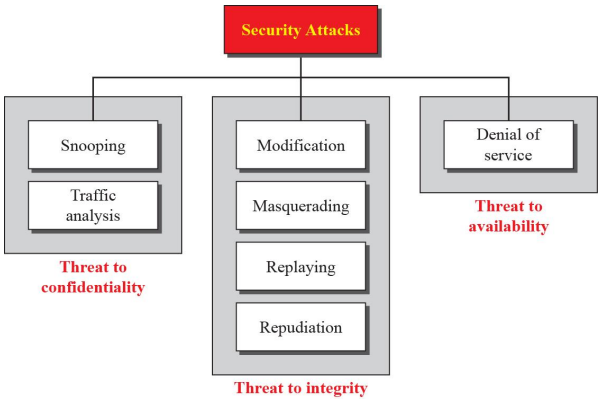
Availability

The third component of information security is availability. The information created and stored by an organization needs to be available to authorized users and applications. Information is useless if it is not available. Information needs to be changed constantly, which means that it must be accessible to those authorized to access it. Unavailability of information is just as harmful to an organization as a lack of confidentiality or integrity. Imagine what would happen to a bank if the customers could not access their accounts for transactions.

16.1.2 Attacks

The three goals of security—confidentiality, integrity, and availability—can be threatened by security attacks. Figure 16.2 relates the taxonomy of attack types to security goals.

Figure 16.1 Taxonomy of attacks with relation to security goals



Attacks threatening confidentiality

In general, two types of attack threaten the confidentiality of information: snooping and traffic analysis. Snooping refers to unauthorized access to or interception of data. Traffic analysis refers other types of information collected by an intruder by monitoring online traffic.

Attacks threatening integrity

The integrity of data can be threatened by several kinds of attack: *modification*, *masquerading*, *replaying*, and *repudiation*.

Attacks threatening availability

We mention only one attack threatening availability: *denial of service*.

16.1.3 Services and Techniques

ITU-T defines some security services to achieve security goals and prevent attacks. Each of these services is designed to prevent one or more attacks while maintaining security goals. The actual implementation of security goals needs some techniques. Two techniques are prevalent today: one is very general (cryptography) and one is specific (steganography).

Cryptography

Some security services can be implemented using cryptography. Cryptography, a word with Greek origins, means "secret writing". However, we use the term to refer to the science and art of transforming messages to make them secure and immune to attacks. Although in the past cryptography referred only to the encryption and decryption of messages using secret keys, today it is defined as involving three distinct mechanisms: symmetric-key encipherment, asymmetric-key encipherment, and hashing. We will discuss all these mechanisms later in the chapter.

Encryption: $C = E_k(P)$ **Decryption: $P = D_k(C)$**

Steganography

Although this chapter and the next are based on cryptography as a technique for implementing security mechanisms, another technique that was used for secret communication in the past is being revived at the present time: steganography. The word steganography, with origins in Greek, means 'covered writing', in contrast with cryptography, which means 'secret writing'. Cryptography means concealing the contents of a message by enciphering; steganography means concealing the message itself by covering it with something else. We leave the discussion of steganography to those books dedicated to this topic.

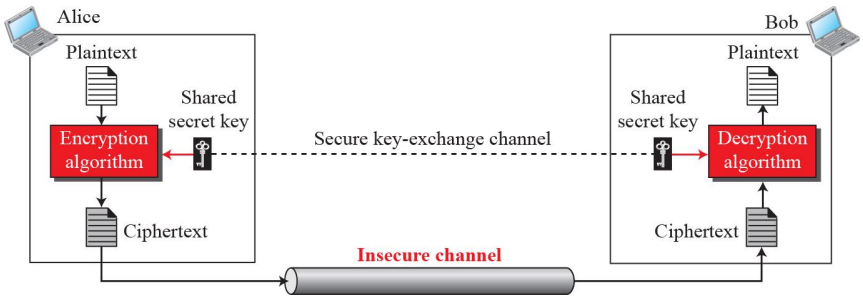
16-2 Confidentiality

We now look at the first goal of security, *confidentiality*. Confidentiality can be achieved using ciphers. Ciphers can be divided into two broad categories: symmetric-key and asymmetric-key.

16.2.1 Symmetric-key Ciphers

A symmetric-key cipher uses the same key for both encryption and decryption, and the key can be used for bidirectional communication, which is why it is called *symmetric*. Figure 16.2 shows the general idea behind a symmetric-key cipher.

Figure 16.2 General idea of symmetric-key cipher



Symmetric-key ciphers are also called secret-key ciphers.

Note that the symmetric-key encipherment uses a single key (the key itself may be a set of values) for both encryption and decryption. In addition, the encryption and decryption algorithms are inverses of each other. If P is the plaintext, C is the ciphertext, and K is the key, the encryption algorithm Ek(x) creates the ciphertext from the plaintext; the decryption algorithm Dk(x) creates the plaintext from the ciphertext. We assume that Ek(x) and Dk(x) are inverses of each other: they cancel the effect of each other if they are applied one after the other on the same input. We have:

Encryption can be thought of as locking the message in a box; decryption can be thought of as unlocking the box. In symmetric-key encipherment, the same key locks and unlocks, as shown in Figure 16.3. Later sections show that the asymmetric-key encipherment needs two keys, one for locking and one for unlocking.

Figure 16.3 Locking and unlocking with the same key

Traditional symmetric-key ciphers

Traditional ciphers belong to the past. However, we briefly discuss them here because they can be thought of as the components of the modern ciphers. To be more exact, we can divide traditional ciphers into *substitution ciphers* and *transposition ciphers*.

A substiturion cipher replaces one symbol with another

Figure 16.4 Representation of plaintext and ciphertext in modulo 26

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

In additive cipher, the plaintext, ciphertext, and key are integer in modulo 26

Example 16.1

Use the additive cipher with key = 15 to encrypt the message "hello".

Solution

We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07	**Encryption: (07 + 15) mod 26**	**Ciphertext: 22 → W**
Plaintext: e → 04	**Encryption: (04 + 15) mod 26**	**Ciphertext: 19 → T**
Plaintext: l → 11	**Encryption: (11 + 15) mod 26**	**Ciphertext: 00 → A**
Plaintext: l → 11	**Encryption: (11 + 15) mod 26**	**Ciphertext: 00 → A**
Plaintext: o → 14	**Encryption: (14 + 15) mod 26**	**Ciphertext: 03 → D**

The ciphertext is therefore "wtaad".

Example 16.2

Use the additive cipher with key 5 = 15 to decrypt the message 'WTAAD'.

Solution

We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W → 22	**Decryption: (22 → 15)**	**mod 26 Plaintext: 07 → h**
Ciphertext: T → 19	**Decryption: (19 → 15)**	**mod 26 Plaintext: 04 → e**
Ciphertext: A → 00	**Decryption: (00 → 15)**	**mod 26 Plaintext: 11 → l**
Ciphertext: A → 00	**Decryption: (00 → 15)**	**mod 26 Plaintext: 11 → l**
Ciphertext: D → 03	**Decryption: (03 → 15)**	**mod 26 Plaintext: 14 → o**

The result is 'hello'. Note that the operation is in modulo 26, which means that we need to add 26 to a negative result (for example -15 becomes 11).

In a *polyalphabetic* cipher, each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many. For example, 'a' could be enciphered as 'D' at the beginning of the text, but as 'N' in the middle. Polyalphabetic ciphers have the advantage of hiding the letter frequency of the underlying language. Eve cannot use single-letter frequency statistics to break the ciphertext.

To see the position dependency of the key, let us discuss a simple polyalphabetic cipher called the autokey cipher. In this cipher, the key is a stream of subkeys, in which each subkey is used to encrypt the corresponding character in the plaintext. The first subkey is a predetermined value secretly agreed upon by Alice and Bob. The second subkey is the value of the first plaintext character (between 0 and 25). The third subkey is the value of the second plaintext character, and so on.

Example 16.3

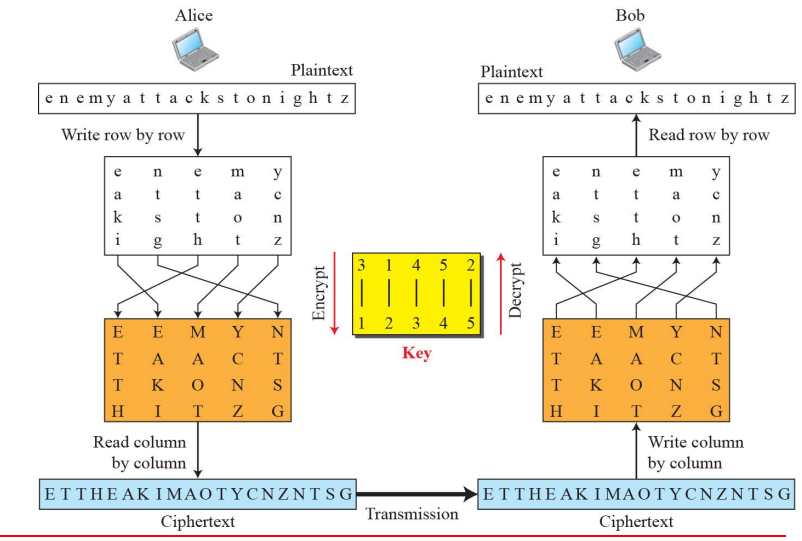
Assume that Alice and Bob agreed to use an autokey cipher with initial key value k1 = 12. Now Alice wants to send Bob the message 'Attack is today'. Enciphering is done character by character. Each character in the plaintext is first replaced by its integer value. The first subkey is added to create the first ciphertext character. The rest of the key is created as the plaintext characters are read. Note that the cipher is polyalphabetic because the three occurrences of 'a' in the plaintext are encrypted differently. The three occurrences of 't' are encrypted differently.

Plaintext:	**a**	**t**	**t**	**a**	**c**	**k**	**i**	**s**	**t**	**o**	**d**	**a**	**y**
P's Values:	**00**	**19**	**19**	**00**	**02**	**10**	**08**	**18**	**19**	**14**	**03**	**00**	**24**
Key stream:	**12**	**00**	**19**	**19**	**00**	**02**	**10**	**08**	**18**	**19**	**14**	**03**	**00**
C's Values:	**12**	**19**	**12**	**19**	**02**	**12**	**18**	**00**	**11**	**7**	**17**	**03**	**24**
Ciphertext:	**M**	**T**	**M**	**T**	**C**	**M**	**S**	**A**	**L**	**H**	**R**	**D**	**Y**

A **transposition cipher** does not substitute one symbol for another; instead it changes the location of the symbols. A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eighth position in the plaintext may appear in the first position of the ciphertext. In other words, a transposition cipher reorders (transposes) the symbols.

A transposition cipher reorders symbols.

Figure 16.5 Transposition cipher



Stream and block ciphers

The literature divides the symmetric ciphers into two broad categories: *stream ciphers* and *block ciphers*.

In a stream cipher, encryption and decryption are done one symbol (such as a character or a bit) at a time. We have a plaintext stream, a ciphertext stream, and a key stream. Call the plaintext stream P, the ciphertext stream C, and the key stream K.

$\mathbf{P = P_1P_2P_3, \ldots}$	$\mathbf{C = C_1C_2C_3, \ldots}$	$\mathbf{K = (k_1, k_2, k_3, \ldots)}$
$\mathbf{C1 = E_{k1}(P1)}$	$\mathbf{C2 = E_{k2}(P2)}$	$\mathbf{C3 = E_{k3}(P3) \ldots}$

Modern symmetric-key ciphers

The literature divides the symmetric ciphers into two broad categories: *stream ciphers* and *block ciphers*.

In a stream cipher, encryption and decryption are done one symbol (such as a character or a bit) at a time. We have a plaintext stream, a ciphertext stream, and a key stream. Call the plaintext stream P, the ciphertext stream C, and the key stream K.

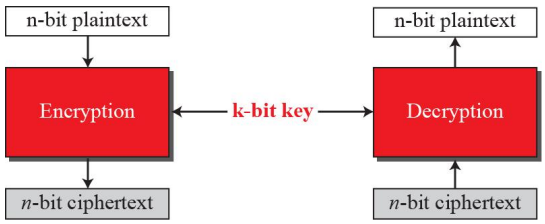
$\mathbf{P = P_1P_2P_3, \ldots}$	$\mathbf{C = C_1C_2C_3, \ldots}$	$\mathbf{K = (k_1, k_2, k_3, \ldots)}$
$\mathbf{C1 = E_{k1}(P1)}$	$\mathbf{C2 = E_{k2}(P2)}$	$\mathbf{C3 = E_{k3}(P3) \ldots}$

Modern symmetric-key ciphers

The traditional symmetric-key ciphers that we have studied so far are character-oriented ciphers. With the advent of the computer, we need bit-oriented ciphers. This is because the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data. It is convenient to convert these types of data into a stream of bits, to encrypt the stream, and then to send the encrypted stream. In addition, when text is treated at the bit level, each character is replaced by 8 (or 16) bits, which means that the number of symbols becomes 8 (or 16) times larger. Mixing a larger number of symbols increases security. A modern cipher can be either a block cipher or a stream cipher.

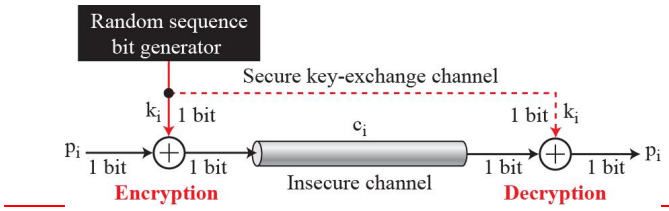
A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key. The decryption algorithm must be the inverse of the encryption algorithm, and both operations must use the same secret key so that Bob can retrieve the message sent by Alice. Figure 16.6 shows the idea behind a modern block cipher.

Figure 16.6 **A modern block cipher**



In addition to modern block ciphers, we can also use modern stream ciphers. The differences between modern stream ciphers and modern block ciphers are similar to the differences between traditional stream and block ciphers. The simplest and the most secure type of synchronous stream cipher is called the one-time pad, which was invented and patented by Gilbert Vernam (Figure 16.7)

Figure 16.7 One-time pad



16.2.2 Asymmetric-key ciphers

Symmetric- and asymmetric-key ciphers will exist in parallel and continue to serve the community. The conceptual differences between the two systems are based on how these systems keep a secret. In symmetric-key cryptography, the secret must be shared between two persons. In asymmetric-key cryptography, the secret is personal (unshared); each person creates and keeps his or her own secret.

In a community of n people, n (n - 1)/2 shared secrets are needed for symmetric-key cryptography; n personal secrets are needed in asymmetric-key cryptography. personal secrets.

Symmetric-key cryptography is based on sharing secrecy; asymmetric-key cryptography is based on personal secrecy.

Whereas symmetric-key cryptography is based on substitution and permutation of symbols (characters or bits), asymmetric-key cryptography is based on applying mathematical functions to numbers. In symmetric-key cryptography, the plaintext and ciphertext are thought of as a combination of symbols. Encryption and decryption permute these symbols or substitute one symbol for another. In asymmetric-key cryptography, the plaintext and ciphertext are numbers; encryption and decryption are mathematical functions that are applied to numbers to create other numbers.

In symmetric-key cryptography, symbols are manipulated: in asymmetric-key cryptography, numbers are manipulated.

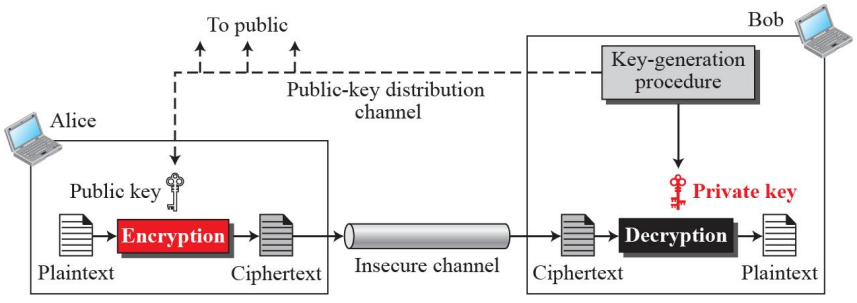
Figure 16.8 Locking and unlocking in asymmetric-key cryptography

Asymmetric-key ciphers are sometimes called public-key ciphers.

General Idea

Figure 16.9 shows the general idea of asymmetric-key cryptography as used for encipherment.

Figure 16.9 General idea of asymmetric-key cryptography

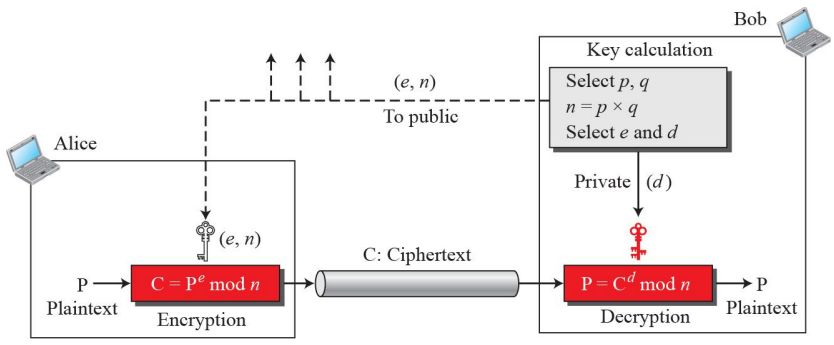


Asymmetric-key cryptography is normally used to encrypt or decrypt small pieces of information..

RSA cryptosystem

Although there are several asymmetric-key cryptosystems, one of the common public-key algorithms is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman). RSA uses two exponents, e and d, where e is public and d is private. Suppose P is the plaintext and C is the ciphertext. Alice uses C 5 Pe mod n to create ciphertext C from plaintext P; Bob uses P 5 Cd mod n to retrieve the plaintext sent by Alice. The modulus n, a very large number, is created during the key generation process.

Figure 16.10 Encryption, decryption, and key generation in RSA



Example 16.4

For the sake of demonstration, let Bob choose 7 and 11 as *p* and *q* and calculate f 5 7 3 11 5 77. The value of f(n) 5 (7 - 1)(11 - 1), or 60. If he chooses *e* to be 13, then *d* is 37. Note that e × d mod 60 = 1. Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5. This system is not safe because p and q are small.

Plaintext: 5
$C = 5^{13} = 26 \bmod 77$
Ciphertext: 26

Ciphertext: 26
$P = 26^{37} = 5 \bmod 77$
Plaintext: 5

Example 16.5

Here is a more realistic example calculated using a computer program in Java. We choosea 512-bit p and q, and calculate n and f(n). We then choose e and calculate d. Finally, we show the results of encryption and decryption. The integer p is a 159-digit number.

```
p =  961303453135835045741915812806154279093098455949 9 6215822583
     150879647940455056470638491257160180347503120986666064924 20
     191808780667421096063354219926661209
```

```
q =  120601919572314469182767942044508960015559250546370339360 61
     798321731482148483764659215389453209175225273226830107120 69
     560460251388714552496900035966004561 7
```

```
n = 115935041739676149688925098646158875237714573754541447754855
    261376147885408326350817276878815968325168468849300625485764
    111250162414552339182927162507656772727460097082714127730 43
    496050055634727456662806009992403710299142447229221577279 85
    317270338393813346926841373276220009666766718318310883734208
    23444370953
```

```
Φ(n)= 115935041739676149688925098646158875237714573754541447754855261 37
      614788540832635081727687881596832516846884930062548576411125016 24
      145523391829271625076567510542336084929167520344826279881175547 87
      657013923444405716989581728196098226361075467211864612171359107 35
      864061400888517026537727726446734106624385766412 8
```

```
e = 35535
d = 580083028600377639360936612896779175946690620896509621804228661113
    805938528223587317062869100300217108590443384021707298690876006115
    306202524959884448047568240966247081485817130463240644077704833134
    010850947385295645071936774061197326557424237217617674620776371642
    076003370853332885321447088595513667029483 1
```

```
P = 190708182608182600261904181 9
```

```
C = 475309123646226827206365550610545180942371796070491716523239243 05
    445296061319932856661784341835911415119741125200568297979457173 60
    361012782188478927415660904800235071907152771859149751884658886 32
    101148354103361657898467968386763733765777465625079280521148141 84
    404814184430812773059004692874248559166462108656
```

```
P = 190708182608182600261904181 9
```

Applications

Although RSA can be used to encrypt and decrypt actual messages, it is very slow if the message is long. RSA, therefore, is useful for short messages. In particular, we will see that RSA is used in digital signatures and other cryptosystems that often need to encrypt a small message without having access to a symmetric key. RSA is also used for authentication, as we will see later in the chapter.

16-3 OTHER ASPECTS OF SECURITY

The cryptography systems that we have studied so far provide confidentiality. However, in modern communication, we need to take care of other aspects of security, such as integrity, message and entity authentication, non-repudiation, and key management. We briefly discuss these issues in this section.

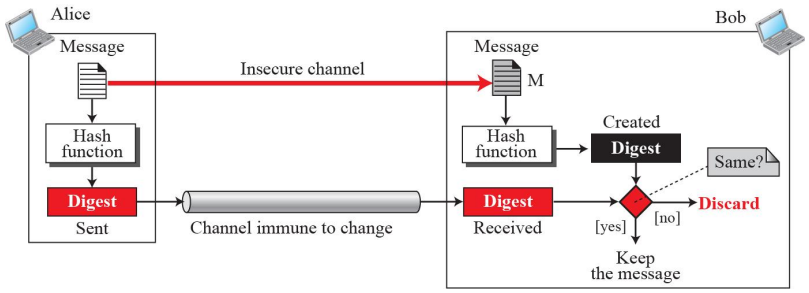
16.3.1 Message integrity

There are occasions where we may not even need secrecy but instead must have integrity: the message should remain unchanged. For example, Alice may write a will to distribute her estate upon her death. The will does not need to be encrypted. After her death, anyone can examine the will. The integrity of the will, however, needs to be preserved. Alice does not want the contents of the will to be changed.

Message and message digest

One way to preserve the integrity of a document is through the use of a *fingerprint*. The electronic equivalent of the document and fingerprint pair is the message and digest pair. To preserve the integrity of a message, the message is passed through an algorithm called a cryptographic hash function. The function creates a compressed image

Figure 16.11 Message and digest



The message digest needs to be safe from change.

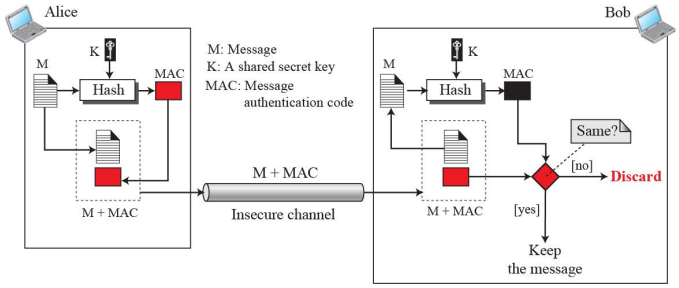
Hash Functions

A cryptographic hash function takes a message of arbitrary length and creates a message digest of fixed length. All cryptographic hash functions need to create a fixed-size digest out of a variable-size message. Creating such a function is best accomplished using iteration. Instead of using a hash function with variable-size input, a function with fixed-size input is created and is used a necessary number of times. The fixed-size input function is referred to as a compression function. It compresses an n-bit string to create an m-bit string where n is normally greater than m. The scheme is referred to as an iterated cryptographic hash function.

16.3.2 Message authentication

A digest can be used to check the integrity of a message. To ensure the integrity of the message and the data origin authentication—that Alice is the originator of the message, not somebody else—we need to include a secret shared by Alice and Bob (that Eve does not possess) in the process; we need to create a message authentication code (MAC). Figure 16.12 shows the idea.

Figure 16.12 Signing the digest



Alice uses a hash function to create a MAC from the concatenation of the key and the message, h (K + M). She sends the message and the MAC to Bob over the insecure channel. Bob separates the message from the MAC. He then makes a new MAC from the concatenation of the message and the secret key. Bob then compares the newly created MAC with the one received. If the two MACs match, the message is authentic and has not been modified by an adversary.

A MAC provides message integrity and message authentication using a combination of a hash function and a secret key.

HMAC

The National Institute of Standards and Technology (NIST) has issued a standard for a nested MAC that is often referred to as HMAC (hashed MAC). The implementation of HMAC is much more complex than the simplified MAC and is not covered in this text.

16.3.3 Digital signature

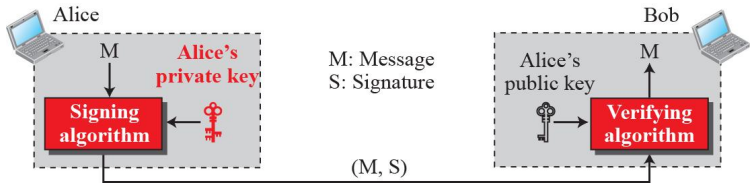
Another way to provide message integrity and message authentication (and some more security services, as we will see shortly) is a digital signature. A MAC uses a secret key to
protect the digest; a digital signature uses a pair of private–public keys.

A digital signature uses a pair of private–public keys.

Process

Figure 16.13 shows the digital signature process. The sender uses a signing algorithm to sign the message. The message and the signature are sent to the receiver. The receiver receives the message and the signature and applies the verifying algorithm to the combination. If the result is true, the message is accepted; otherwise, it is rejected.

Figure 16.13 Digital signature process



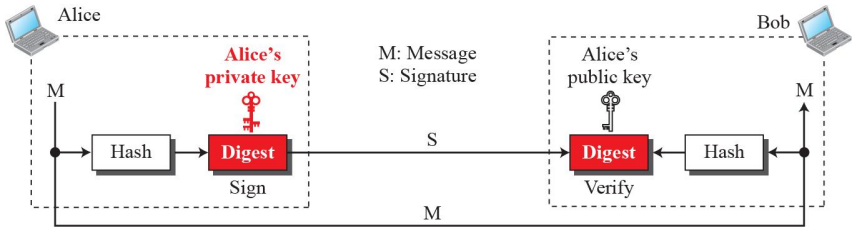
A digital signature needs a public-key system. The signer signs with her private key; the verifier verifies with the signer's public key.

A cryptosystem uses the private and public keys of the receiver; a digital signature uses the private and public keys of the sender.

Signing the digest

In a digital signature system, the messages are normally long, but we have to use asymmetric-key schemes. The solution is to sign a digest of the message, which is much shorter than the message. A carefully selected message digest has a one-to-one relationship with the message. The sender can sign the message digest and the receiver can verify the message digest. The effect is the same. Figure 16.14 shows signing a digest in a digital signature system.

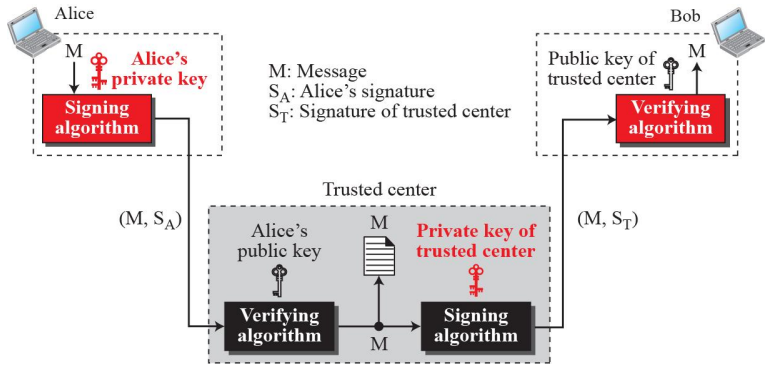
Figure 16.14 Signing the digest



Services

We discussed several security services including message confidentiality, message authentication, message integrity, and non-repudiation. A digital signature can directly provide the last three; for message confidentiality we still need encryption/decryption.

Figure 16.15 Using a trusted center for nonrepudiation



16.3.4 Entity authentication

Entity authentication is a technique designed to let one party verify the identity of another party. An entity can be a person, a process, a client, or a server. The entity whose identity needs to be proven is called the claimant; the party that tries to verify the identity of the claimant is called the verifier.

Verification categories

In entity authentication, the claimant must identify him- or herself to the verifier. This can be done with one of three kinds of witnesses: something known, something possessed, or something inherent.

- **Something known**
- **Something possessed**
- **Something inherent**

Entity versus message authentication

There are two differences between entity authentication and message authentication (data-origin authentication).

1. Message authentication (or data-origin authentication) might not happen in real time; entity authentication does.
2. Message authentication simply authenticates one message; the process needs to be repeated for each new message. Entity authentication authenticates the claimant for the entire duration of a session.

Passwords

The simplest and oldest method of entity authentication is the use of a password, which is something that the claimant knows. A password is used when a user needs to access a system's resources (login). Each user has a user identification that is public, and a password that is private. Passwords, however, are very prone to attack. A password can be stolen, intercepted, guessed, and so on.

Challenge-responses

In challenge-response authentication, the claimant proves that she knows a secret without sending it to the verifier.

In challenge-response authentication, the claimant proves that she knows a secret without sending it to the verifier.

Figure 16.16 Unidirectional, symmetric-key authentication

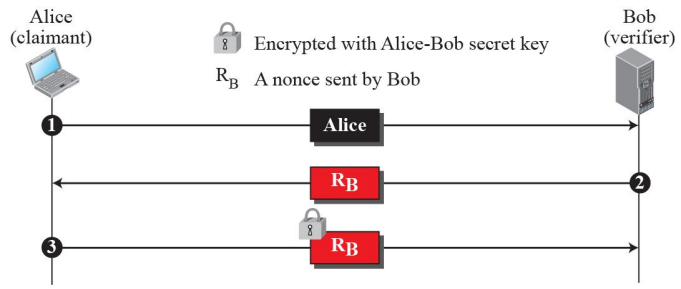


Figure 16.17 Unidirectional, asymmetric-key authentication

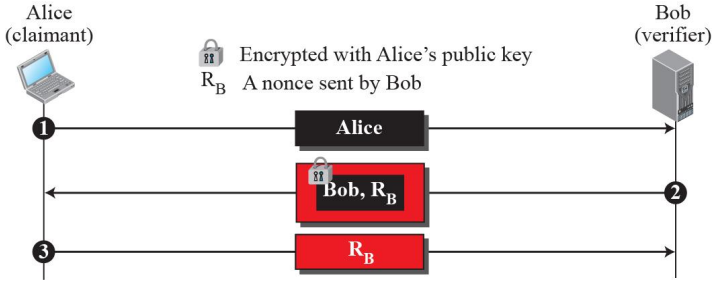
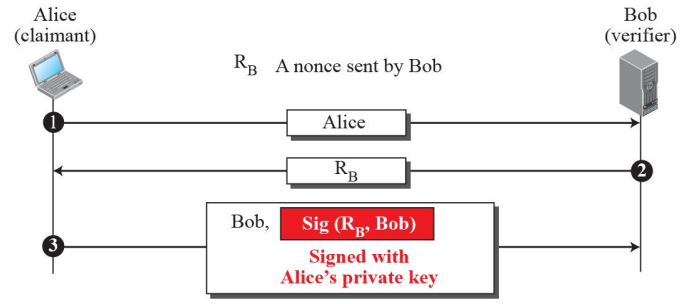


Figure 16.18 Digital signature, unidirectional authentication



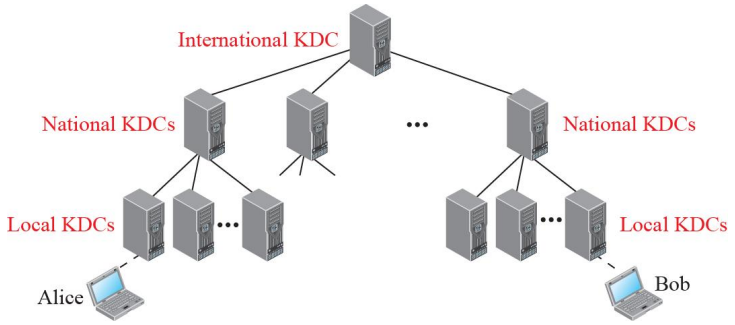
16.3.5 Key management

We discussed symmetric-key and asymmetric-key cryptography in the previous sections. However, we have not yet discussed how secret keys in symmetric-key cryptography, and public keys in asymmetric-key cryptography, are distributed and maintained. This section touches on these two issues.

Symmetric-key distribution

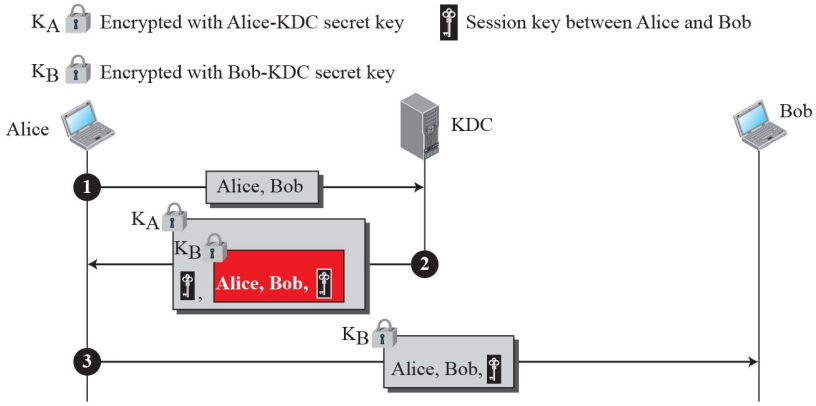
Symmetric-key cryptography is more efficient than asymmetric-key cryptography for enciphering large messages. Symmetric-key cryptography, however, needs a shared secret key between two parties.

Figure 16.19 Multiple KDCs



A session symmetric key between two parties is used only once.

Figure 16.20 Creating a session key using KDC

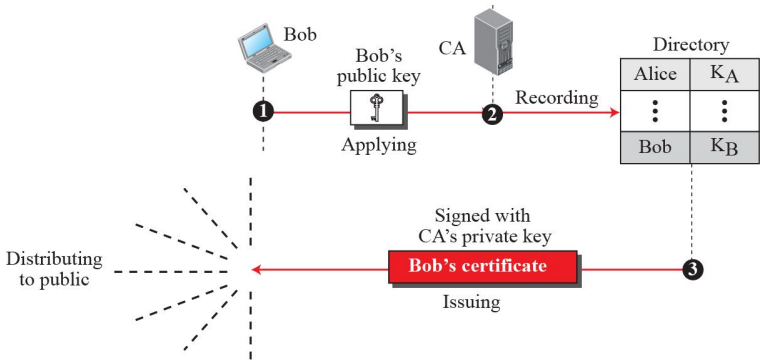


Public-key distribution

In asymmetric-key cryptography, people do not need to know a symmetric shared key. If Alice wants to send a message to Bob, she only needs to know Bob's public key, which is open to the public and available to everyone. If Bob needs to send a message to Alice, he only needs to know Alice's public key, which is also known to everyone. In public-key cryptography, everyone shields a private key and advertises a public key.

In public-key cryptography, everyone has access to everyone's public key; public keys are available to the public.

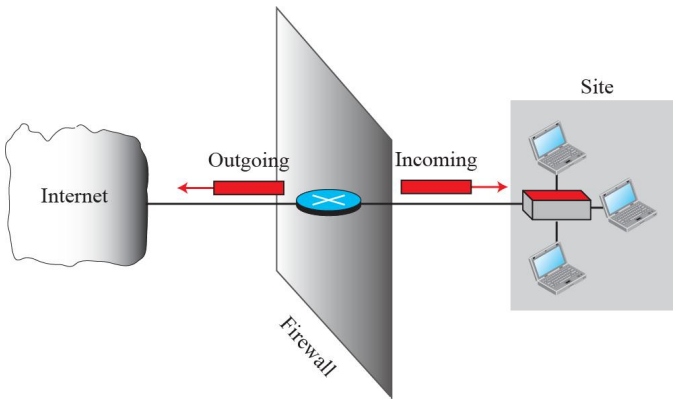
Figure 16.21 Certification authority



16-4 FIREWALS

All previous security measures cannot prevent Eve from sending a harmful message to a system. To control access to a system we need firewalls. A firewall is a device (usually a router or a computer) installed between the internal network of an organization and the rest of the Internet. It is designed to forward some packets and filter (not forward) others. Figure 16.22 shows a firewall.

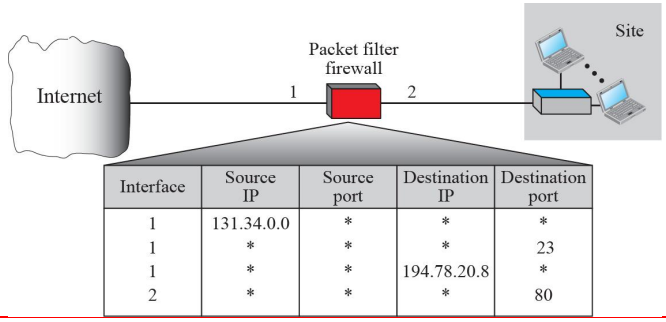
Figure 16.22 General idea of symmetric-key cipher



16.4.1 Packet filter firewall

A firewall can be used as a packet filter. It can forward or block packets based on the information in the network-layer and transport-layer headers: source and destination IP addresses, source and destination port addresses, and type of protocol (TCP or UDP). A packet-filter firewall is a router that uses a filtering table to decide which packets must be discarded (Figure 16.23).

Figure 16.23 Packet-filter firewall



Interface	Source IP	Source port	Destination IP	Destination port
1	131.34.0.0	*	*	*
1	*	*	*	23
1	*	*	194.78.20.8	*
2	*	*	*	80

16.4.1 Proxy firewall

The packet-filter firewall is based on the information available in the network-layer and transport-layer headers (IP and TCP/UDP). However, sometimes we need to filter a message based on the information available in the message itself (at the application layer).

A proxy firewall stands between the customer computer and the corporation computer. When the user client process sends a message, the application gateway runs a server process to receive the request. The server opens the packet at the application level and finds out if the request is legitimate. If it is, the server acts as a client process and sends the message to the real server in the corporation. If it is not, the message is dropped and an error message is sent to the external user. Figure 16.24 shows a proxy firewall.

Figure 16.24 Proxy firewall