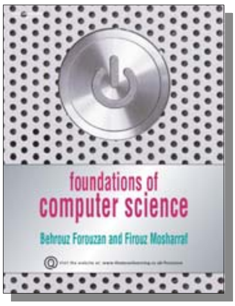
16

Security



Source: Foundations of Computer Science © Cengage Learning

Objectives

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

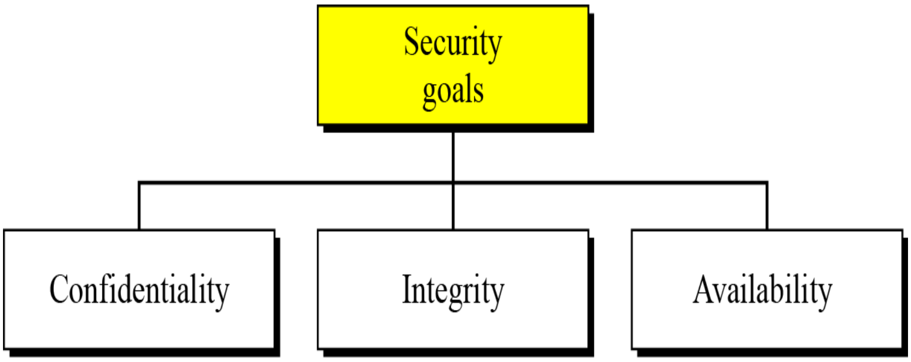
- ❑ Define three security goals— confidentiality, integrity and availability —as well as attacks that threatens these security goals.
- ❑ Define five security services to prevent security attacks— data confidentiality, data integrity, authentication, non-repudiation and access control.
- ❑ Discuss two techniques for providing security services: cryptography and steganography.
- ❑ Distinguish between symmetric-key cryptography and asymmetric-key cryptography and show how confidentiality can be provided using either symmetric-key or asymmetric-key ciphers.
- ❑ Show how integrity can be provided using cryptographic hashing functions.
- ❑ Discuss the idea of digital signatures and how they can provide message integrity, message authentication and non-repudiation.
- ❑ Briefly discuss entity authentication and categories of witnesses: something known, something possessed and something inherent.
- ❑ Discuss four techniques used for entity authentication: password-based, challenge-response, zero knowledge and biometrics.
- ❑ Discuss key management.

16-1 INTRODUCTION

In this section we describe the general idea behind information security.

Security goals

We will first discuss **three security goals**: ***confidentiality***, ***integrity*** and **availability** (Figure 16.1).



Security goals

Confidentiality

Integrity

Availability

Figure 16.1 **Taxonomy of security goals**

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.

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**.

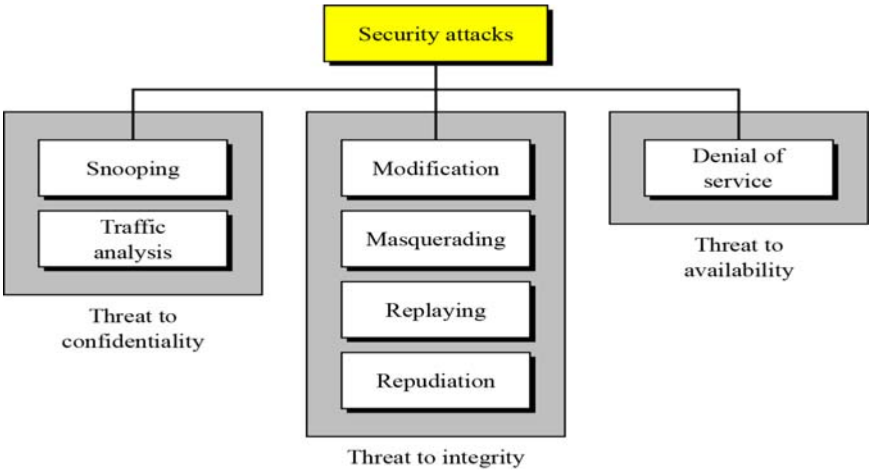


Figure 16.2 **Taxonomy of attacks with relation to security goals**

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

Denial of service (DoS) attacks **may slow down or totally interrupt the service of a system**. The attacker can use several strategies to achieve this. They might make the system so busy that it collapses, or they might intercept messages sent in one direction and make the sending system believe that one of the parties involved in the communication or message has lost the message and that it should be resent.

Security services

Standards have been defined for security services to **achieve security goals and prevent security attacks**. Figure 16.3 shows the taxonomy of the **five common services**.

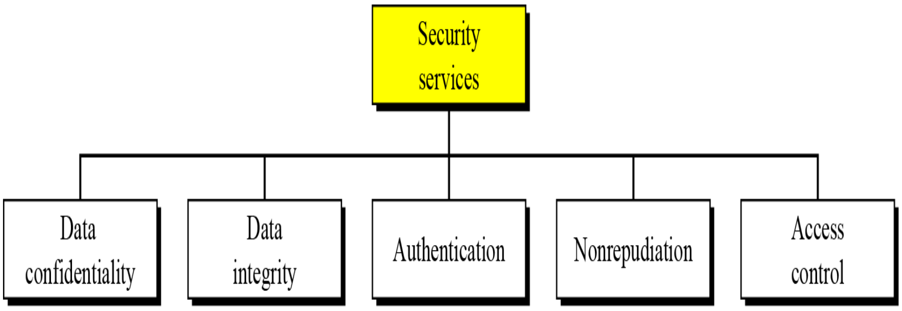


Figure 16.3 **Security services**

Techniques

The actual implementation of security goals needs some help from mathematics. **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**".

Steganography

The word steganography, with its origin in Greek, means "**covered writing**", in contrast to cryptography, which means "secret writing".

16-2 SYMMETRIC-KEY CRYPTOGRAPHY

Figure 16.4 shows the general idea behind symmetric-key cryptography. **Alice can send a message to Bob over an insecure channel** with the assumption that an adversary, Eve, cannot understand the contents of the message by simply eavesdropping on the channel.

The original message from Alice to Bob is referred to as plaintext; the message that is sent through the channel is referred to as the ciphertext. **Alice uses an encryption algorithm and a shared secret key**. **Bob uses a decryption algorithm and the same secret key**.

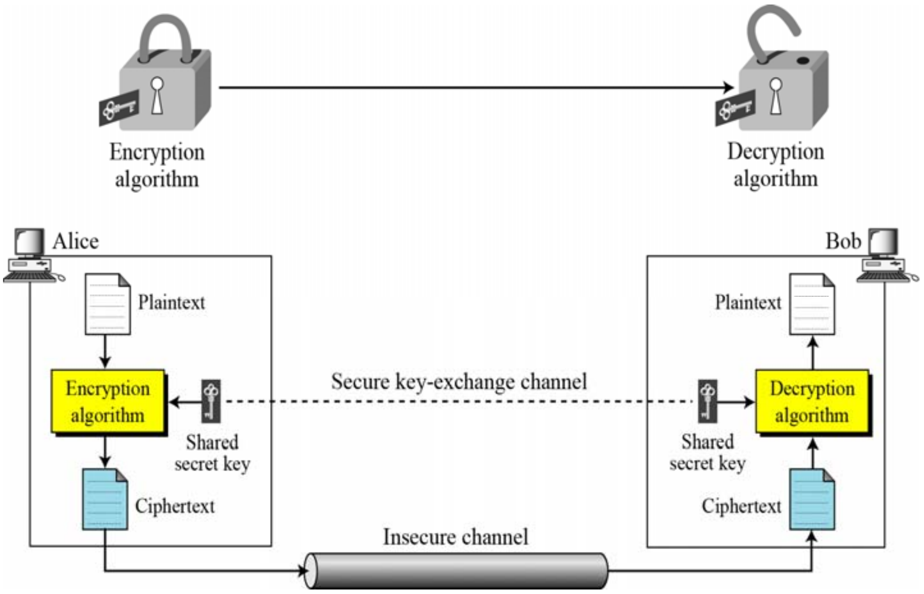


Figure 16.4 **The general idea of symmetric-key cryptography**

16.13

Traditional ciphers

Traditional ciphers used **two techniques for hiding information from an intruder**: *substitution* and *transposition*.

Substitution ciphers

A substitution cipher **replaces one symbol with another**. If the symbols in the plaintext are alphabetic characters, we replace one character with another.

A substitution cipher replaces one symbol with another.

The simplest substitution cipher is a **shift cipher** (additive cipher).

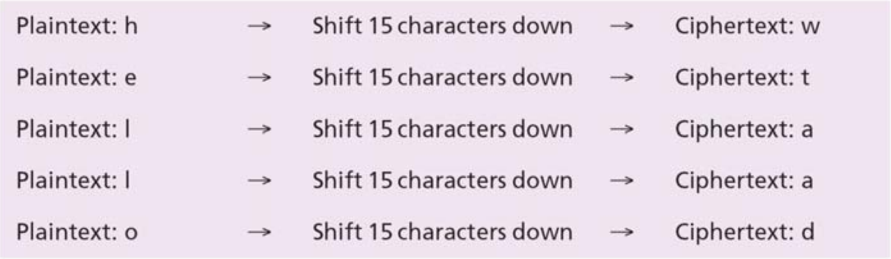
16.14

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	→	Shift 15 characters down	→	Ciphertext: w
Plaintext: e	→	Shift 15 characters down	→	Ciphertext: t
Plaintext: l	→	Shift 15 characters down	→	Ciphertext: a
Plaintext: l	→	Shift 15 characters down	→	Ciphertext: a
Plaintext: o	→	Shift 15 characters down	→	Ciphertext: d

The ciphertext is therefore “wtaad”.

Transposition ciphers

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, while 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.

Example 16.2

Alice needs to send the message "**Enemy attacks tonight**" to Bob. Alice and Bob have agreed to divide the text into groups of five characters and then permute the characters in each group. The following shows the grouping after adding a bogus character (z) at the end to make the last group the same size as the others.

enemy attac kston ightz

The key used for encryption and decryption is a permutation key, which shows how the character are permuted. For this message, assume that Alice and Bob used the following key:

Encryption ↓	3	1	4	5	2	↑ Decryption
	1	2	3	4	5	

eemyn taact tkons hitzg

Example 16.2 Continued

The third character in the plaintext block becomes the first character in the ciphertext block, the first character in the plaintext block becomes the second character in the ciphertext block and so on. The permutation yields:

eemyn taact tkons hitzg

Alice sends the ciphertext "eemyntaacttkonshitzg" to Bob. Bob divides the ciphertext into five-character groups and, using the key in the reverse order, finds the plaintext.

Modern symmetric-key ciphers

Since traditional ciphers are no longer secure, modern symmetric-key ciphers have been developed during the last few decades. Modern ciphers normally use a combination of substitution, transposition and some other complex transformations to create a ciphertext from a plaintext. Modern ciphers are bit-oriented (instead of character-oriented). The plaintext, ciphertext and the key are strings of bits. In this section we briefly discuss **two examples of modern symmetric-key ciphers: DES and AES**. The coverage of these two ciphers is short: interested readers can consult the references at the end of the chapter for more details.

DES

The **Data Encryption Standard (DES)** is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in **1977**. DES has been the most widely used symmetric-key block cipher since its publication (Figure 16.5).

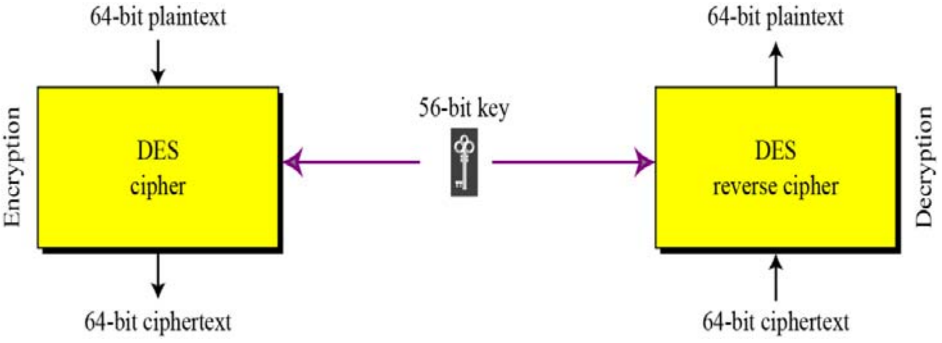


Figure 16.5 **The general design of the DES encryption cipher**

AES

The **Advanced Encryption Standard (AES)** is a symmetric-key block cipher published by the US National Institute of Standards and Technology (NIST) in **2001** in response to the shortcoming of DES, for example its small key size. See Figure 16.6.

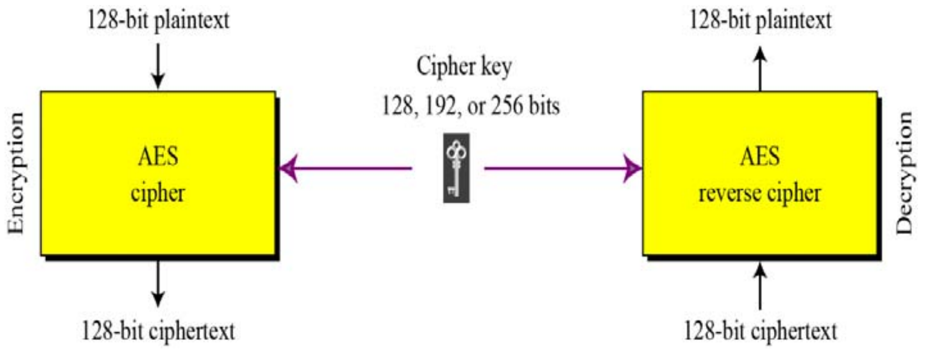


Figure 16.6 **Encryption and decryption with AES**

16-3 ASYMMETRIC-KEY CRYPTOGRAPHY

Figure 16.7 shows the general idea of asymmetric-key cryptography as used for confidentiality. The figure shows that, unlike symmetric-key cryptography, there are distinctive keys in asymmetric-key cryptography: a **private key** and a **public key**. If encryption and decryption are thought of as locking and unlocking padlocks with keys, then the padlock that is locked with a public key can be unlocked only with the corresponding private key. Eve should not be able to advertise her public key to the community pretending that it is Bob’s public key.

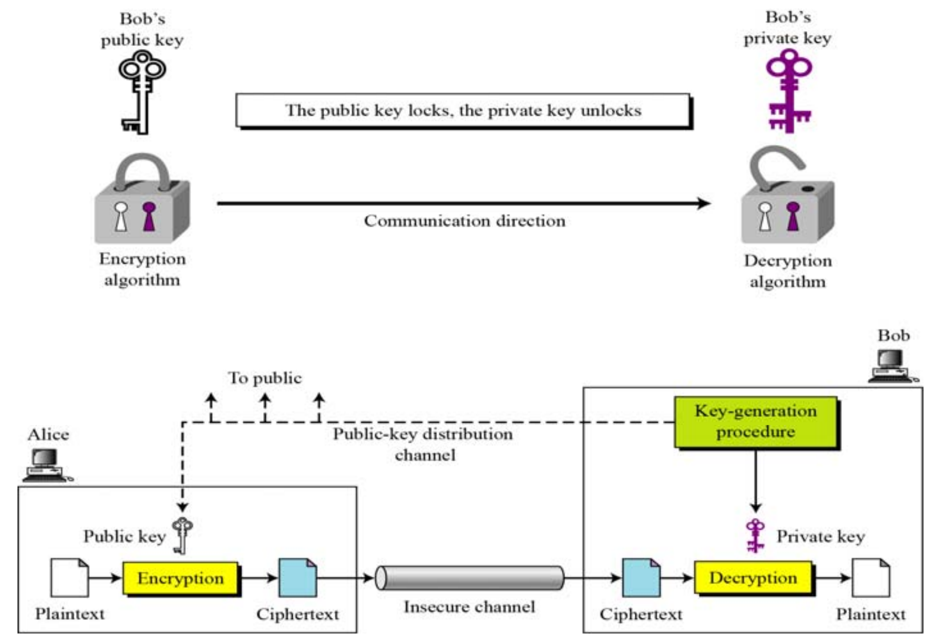


Figure 16.7 **The general idea behind asymmetric-key cryptography**

16.23

Encryption/Decryption

The **ciphertext can be thought of as $C = f\ (K_{public}, P)$ and the plaintext can be thought of as $P = g\ (K_{private}, C)$** in which *f* and *g* are mathematical functions.

The most common public-key algorithm is the **RSA algorithm**, named after its inventors, Ron **R**ivest, Aid **S**hamir, and Leonard **A**dleman.

In RSA algorithm, two prime numbers *p* and *q* are used for creating a modulus $n = p \times q$. Then calculate two exponents *e* and *d*. The public key is (*n* and *e*) and its private key is (*d*).

Encryption: $C = P^e \bmod n$

Decryption: $P = C^d \bmod n$

Example 16.3

Bob chooses p = 7 and q = 11 and calculates $n = 7 \times 11 = 77$. Now he chooses two exponents, e = 13 and d = 37, using the complex process mentioned before. The public key is (n = 77 and e = 13) and the private key is (d = 37). Now imagine that Alice wants to send the plaintext 5 to Bob. The following shows the encryption and decryption.

Encryption at Alice's site

P:5 => C = P^e mod n

= 5^{13} mod 77 = 26

Decryption at Bob's site

C:26 => P = C^d mod n

= 26^{37} mod 77 = 5

16-4 COMPARISON OF METHODS

Both symmetric-key and asymmetric-key cryptography will continue to exist in parallel. We believe that they are complements of each other: the advantages of one can compensate for the disadvantages of the other.

The number of secrets

The conceptual differences between the two systems are based on how these systems keep a secret. **In symmetric-key cryptography,** the secret token must be shared between two parties. **In asymmetric-key cryptography**, the token is unshared: each party creates its own token.

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

16.27

A need for both systems

There are other aspects of security besides confidentiality that need asymmetric-key cryptography. These include authentication and digital signatures (discussed later). Whereas **symmetric-key cryptography** is based on substitution and permutation of symbols, **asymmetric-key cryptography** is based on applying mathematical functions to numbers.

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

16.28

16-5 OTHER SECURITY SERVICES

The cryptography systems we have studied so far provide secrecy, or confidentiality, but none of the other services we discussed at the beginning of the chapter. In this section, we show how we can create other services.

Message integrity

There are occasions on which we may not even need secrecy, but instead must have integrity. One way to preserve the integrity of a document was traditionally 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 of the message that can be used like a fingerprint. Figure 16.8 shows the message, cryptographic hash function and message digest.

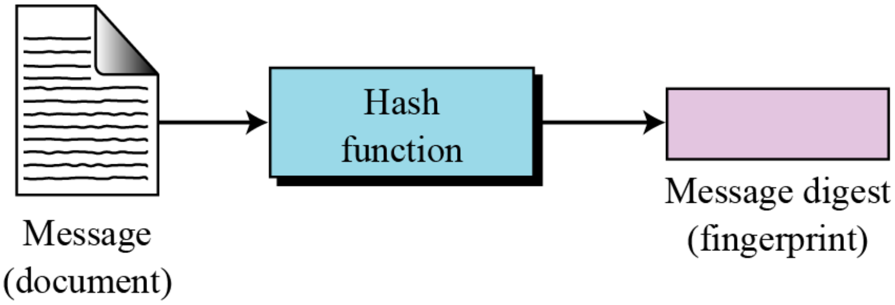


Figure 16.8 Message and digest

The message digest needs to be safe from change.

Checking integrity

To check the integrity of a message or document, we run the cryptographic hash function again and compare the new message digest with the previous one. If both are the same, we are sure that the original message has not been changed. Figure 16.9 shows the idea.

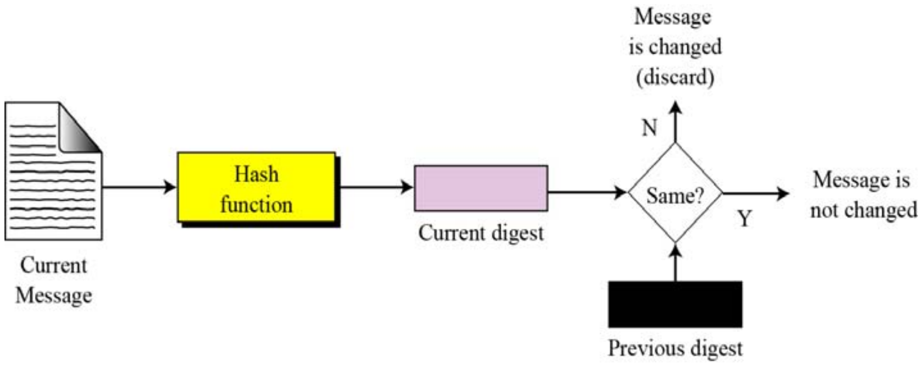


Figure 16.9 Checking integrity

Message authentication

A message digest guarantees the integrity of a message—it guarantees that the message has not been changed. A message digest, however, does not authenticate the sender of the message. When Alice sends a message to Bob, Bob needs to know that the message is really from Alice. To provide message authentication, Alice needs to provide proof that it is she who is sending the message and not an impostor. A message digest per se cannot provide such a proof. The digest created by a cryptographic hash function is normally called a **modification detection code (MDC)**. What we need for message authentication is a **message authentication code (MAC)**.

16.33

Message authentication code (MAC)

To ensure the integrity of the message and authenticate its origin, we need to change an MDC to a MAC. The difference is that the latter includes a secret between Alice and Bob.

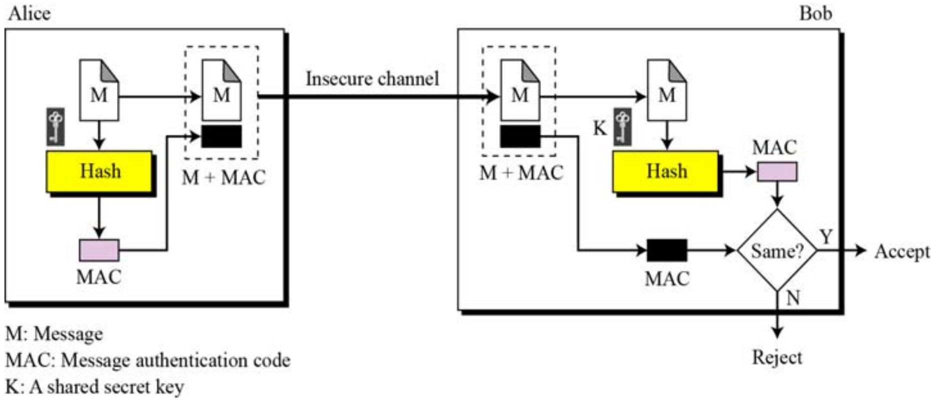


Figure 16.10 **Message authentication code**

16.34

Digital signatures

We are all familiar with the concept of a signature. A person signs a document to show that it originated from him/her or was approved by him/her. The signature is proof to the recipient that the document comes from the correct entity. In other words, a signature on a document, when verified, is a sign of authentication—the document is authentic. When Alice sends a message to Bob, Bob needs to check the authenticity of the sender: he needs to be sure that the message comes from Alice and not Eve. Bob can ask Alice to sign the message electronically. In other words, an electronic signature can prove the authenticity of Alice as the sender of the message. We refer to this type of signature as a **digital signature**.

16.35

Digital signature process

Figure 16.11 shows the digital signature process. The sender uses a **signing algorithm** to sign the message. The message and the signature are sent to the recipient. The recipient 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.

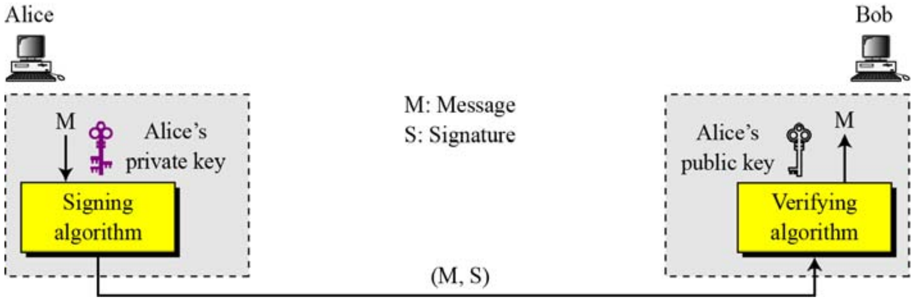


Figure 16.11 **The digital signature process**

16.36

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 recipient: a digital signature uses the private and public keys of the sender.

Signing the digest

Asymmetric-key cryptosystems are very inefficient when dealing with long messages. 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 itself.

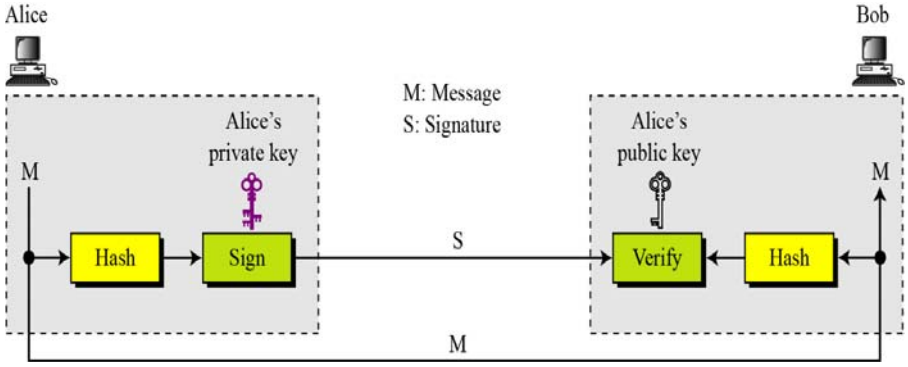


Figure 16.12 **Signing the digest**

Services

A digital signature provides three out of our initial five security services: **message authentication**, **message integrity** and **non-repudiation**. We have seen the first two, the third can be done using the following figure.

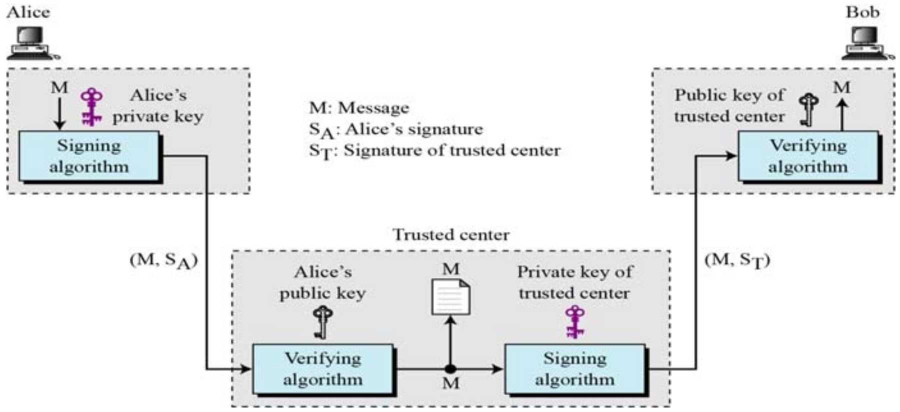


Figure 16.13 **Non-repudiation using digital signatures**

Entity authentication

Entity authentication is a technique designed to let one party prove 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 proved is called the **claimant**: the party that tries to prove the identity of the claimant is called the **verifier**.

Data-origin versus entity authentication

There are two differences between message authentication (data-origin authentication), discussed before, and entity authentication, discussed in this section.

- ❑ Message authentication (or data-origin authentication) might not happen in real time, while entity authentication does.
- ❑ 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.

Verification categories

In entity authentication, the claimant must identify themselves to the verifier. This can be done with one of three kinds of witnesses:

- ❑ **Something known**. This is a secret known only by the claimant that can be checked by the verifier. Examples are a password, a PIN, a secret key and a private key.
- ❑ **Something possessed**. This is something that can prove the claimant’s identity. Examples are a passport, a driver’s license, an identification card and a credit card
- ❑ **Something inherent**. This is an inherent characteristic of the claimant. Examples are conventional signatures, fingerprints, voice, facial characteristics, retinal pattern and handwriting.

16-6 KEY MANAGEMENT

To use **symmetric-key cryptography**, a shared secret key needs to be established between the two parties. To use **asymmetric-key cryptography**, each entity needs to create a pair of keys and distribute the public key securely to the community. Key management defines some procedures to create and distribute keys securely.

Symmetric-key distribution

In a community with n entities, $n(n-1)/2$ keys are needed for symmetric-key communication. The number of keys is not the only problem: the distribution of keys is another. If Alice and Bob want to communicate, they need a way to exchange a secret key. If Alice wants to communicate with a million people, how can she exchange a million keys with them? Using the Internet is definitely not a secure method. It is obvious that we need an efficient way to maintain and distribute secret keys.

Key distribution center: KDC

A practical solution is the use of a trusted third party, referred to as a **key-distribution center (KDC).** Each person establishes a shared secret key with the KDC. A secret key is established between the KDC and each member. The process is as follows:

1. Alice sends a request to the KDC stating that she needs a session (temporary) secret key between herself and Bob.
2. The KDC informs Bob about Alice’s request.
3. If Bob agrees, a session key is created between the two.

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

Public-key distribution

In asymmetric-key cryptography, people do not need 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.

Public announcement

The naive approach is to announce public keys publicly. Bob can put his public key on his web site or announce it in a local or national newspaper. When Alice needs to send a confidential message to Bob, she can obtain Bob's public key from his site or from the newspaper, or even send a message to ask for it. This approach, however, is not secure—it is subject to forgery.

Trusted center

A more secure approach is to have a trusted center retain a directory of public keys. The directory, like the one used in a telephone system, is dynamically updated. Each user can select a private and public key, keep the private key, and deliver the public key for insertion into the directory. The center requires that each user register in the center and prove their identity. The directory can be publicly advertised by the trusted center. The center can also respond to any inquiry about a public key.

Certification authority

The previous approach can create a heavy load on the center if the number of requests is large. The alternative is to create **public-key certificates**. Bob wants two things: he wants people to know his public key, and he wants no-one to accept a forged public key as his. Bob can go to a **certification authority (CA)**, a government authority that binds a public key to an entity and issues a certificate. The CA itself has a well known public key that cannot be forged. The CA issues a certificate for Bob. To prevent the certificate itself from being forged, the CA signs the certificate with its private key. Now Bob can upload the signed certificate. Anyone who wants Bob's public key downloads the signed certificate and uses the center's public key to extract Bob's public key.