Network Security

Advances in the field of computer networks have made information security even more important. Computer systems have to be equipped with mechanisms for securing data. Computer networks need provisions that secure data from possible intrusions. Security is especially crucial in wireless networks, as a wireless medium, by its nature, is vulnerable to intrusions and attacks. This chapter focuses on *network security*. The following major topics in network security are covered.

- *Overview of network security: elements and threats*
- *Overview of security methods*
- *Secret-key encryption protocols*
- *Public-key encryption protocols*
- *Authentication: message digest and digital signatures*
- *Security of IP and wireless networks*
- *Firewalls*

We begin by identifying and classifying types of network threats, hackers, and attacks, including DNS hacking attacks and router attacks. Network security can be divided into two broad categories: *cryptographic techniques* and *authentication techniques* (verification). Both secret-key and public-key encryption protocols are presented. We then discuss message authentication and digital signature methods, through which a receiver can be assured that an incoming message is from whom it says it is.

OVERVIEW OF NETWORK SECURITY

Network security is a top-priority issue in data networks. As communication networks are growing rapidly, security issues have pushed to the forefront of concern for end users, administrators, and equipment suppliers. Despite enormous joint efforts by various groups to develop effective security solutions for networks, hackers continue to pose new, serious threats by taking advantage of weaknesses present in the Internet infrastructure.

Elements of Network Security

Network security is concerned mainly with the following two elements:

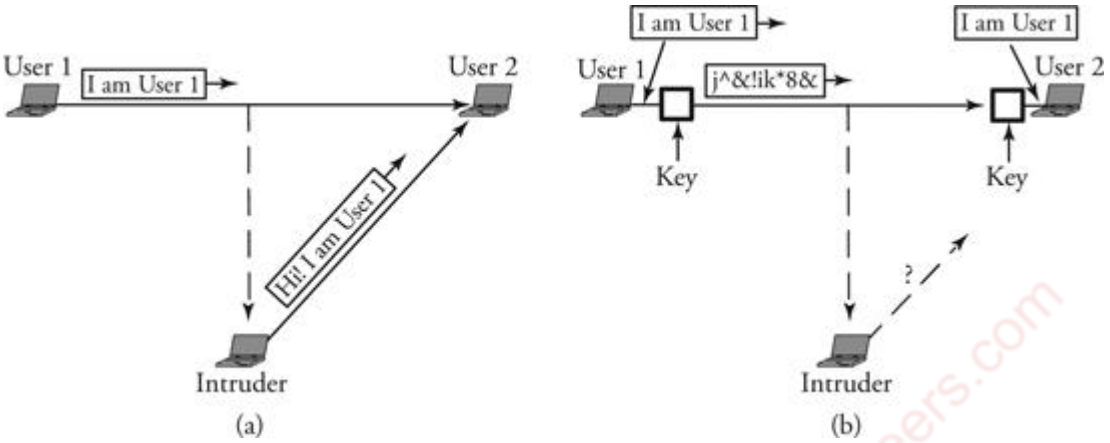
1. *Confidentiality*. Information should be available only to those who have rightful access to it.

2. *Authenticity and integrity*. The sender of a message and the message itself should be verified at the receiving point.

In Figure 10.1, user 1 sends a message (“I am user 1”) to user 2. In part (a) of the figure, the network lacks any security system, so an intruder can receive the message, change its content to a different message (“Hi! I am user 1”) and send it to user 2. User 2 may not know that this falsified message is

really from user 1 (authentication) and that the content of the message is what user 1 (confidentiality). In part (b) of the figure, a security block is added to each side of the communication, and a secret key that only users 1 and 2 would know about is included. Therefore, the message is changed to a form that cannot be altered by the intruder, who would be disabled in this communication transaction.

Figure 10.1 (a) Message content and sender identity falsified by intruder; (b) a method of applied security



In general, no protocol or network architecture can ensure full security. Internet routing is based on a distributed system of many routers, switches, and protocols. These protocols have a number of points of vulnerabilities that can be exploited to cause such problems as misdelivery or nondelivery of user traffic, misuse of network resources, network congestion and packet delays, and the violation of local routing policies.

Threats to Network Security

Internet infrastructure *attacks* are broadly classified into four categories, as follows:

1. DNS hacking
2. Routing table poisoning
3. Packet mistreatment
4. Denial of service

Among these threats, the first three attacks are related to network infrastructure; *denial-of-service attacks* are related to end systems.

DNS Hacking Attacks

As mentioned in Chap As mentioned in Chapter 9, the *Domain Name System* (DNS) server is a distributed hierarchical and global directory that translates domain names into numerical IP address. DNS is a critical infrastructure, and all hosts contact DNS to access servers and start connections. In the normal mode of operation, hosts send UDP queries to the DNS server. Servers reply with a proper answer, or direct the queries to smarter servers. A DNS server also stores information other than host addresses.

Name-resolution services in the modern Internet environment are essential for e-mail transmission, navigation to Web sites, or data transfer. Thus, an attack on DNS can potentially affect a large portion of the Internet. A DNS *hacking attack* may result in the lack of data authenticity and integrity and can appear in any of the following forms:

1. An *information-level attack* forces a server to correspond with other than the correct answer. With cache poisoning, a hacker tricks a remote name server into caching the answer for a third-party domain by providing malicious information for the domain's authorized servers. Hackers can then redirect traffic to a preselected site.

2. In a *masquerading attack*, the adversary poses as a trusted entity and obtains all the secret information. In this guise, the attacker can stop any message from being transmitted further or can change the content or redirect the packet to bogus servers. This action is also known as a *middle-man attack*.

3. The attacker normally sends queries to each host and receives in reply the DNS host name. In an *information leakage attack*, the attacker sends queries to all hosts and identifies which IP addresses are not used. Later on, the intruder can use those IP addresses to make other types of attacks.

4. Once a domain name is selected, it has to be registered. Various tools are available to register domain names over the Internet. If the tools are not smart enough, an invader might obtain secure information and use it to highjack the domain later. In the *domain highjacking attack*, whenever a user enters a domain address, she/he is forced to enter into the attacker's Web site. This can be very irritating and can cause a great loss of Internet usage ability.

Routing Table Poisoning Attacks

A *routing table poisoning attack* is the undesired modification of routing tables. An attacker can do this by maliciously modifying the routing information update packets sent by routers. This is a challenging and important problem, as a routing table is the basis of routing in the Internet. Any false entry in a routing table could lead to significant consequences, such as congestion, an overwhelmed host, looping, illegal access to data, and network partition. Two types of routing table poisoning attacks are the *link attack* and the *router attack*.

A *link attack* occurs when a hacker gets access to a link and thereby intercepts, interrupts, or modifies routing messages on packets. Link attacks act similarly on both the link-state and the distance-vector protocols discussed in Chapter 7. If an attacker succeeds in placing an attack in a link-state routing protocol, a router may send incorrect updates about its neighbours or remain silent even if the link state of its neighbor has changed. The attack through a link can be so severe that the attacker can program a router to either drop packets from a victim or readdress packets to a victim, resulting in a lower throughput of the network. Sometimes, a router can stop an intended packet from being forwarded further. However, since more than one path to any destination exists, the packet ultimately reaches its destination.

Router attacks may affect the link-state protocol or even the distance-vector protocol. If link-state protocol routers are attacked, they become malicious. They may add a nonexisting link to a routing table, delete an existing link, or even change the cost of a link. This attack may cause a router to simply ignore the updates sent by its neighbors, leading to a serious impact on the operability of the network traffic flow.

In the distance-vector protocol, an attacker may cause routers to send wrong updates about any node in the network, thereby misleading a router and resulting in network problems.

Most unprotected routers have no way to validate updates. Therefore, both link-state and distance-vector router attacks are very effective. In the distance-vector protocol, for example, a malicious router can send wrong information in the form of a distance vector to all its neighbors. A neighbor may not be able to detect this kind of attack and thus proceeds to update its routing table, based on wrong distance vectors. The error can in turn be propagated to a great portion of the network before being detected.

Packet-Mistreatment Attacks

A *packet-mistreatment attack* can occur during any data transmission. A hacker may capture certain data packets and mistreat them. This type of attack is very difficult to detect. The attack may result in congestion, lowering throughput, and denial-of-service attacks. Similar to routing table poisoning attacks, packet-mistreatment attacks can also be subclassified into *link attacks* and *router attacks*. The link attack causes interruption, modification, or replication of data packets. A router attack can misroute all packets and may result in congestion or denial of service. Following are some examples of a packet-mistreatment attack:

• *Interruption*. If an attacker intercepts packets, they may not be allowed to be propagated to their destinations, resulting in a lower throughput of the network. This kind of attack cannot be detected easily, as even in normal operations, routers can drop some packets, for various reasons.

• *Modification*. Attackers may succeed in accessing the content of a packet while in transit and change its content. They can then change the address of the packet or even change its data. To solve this kind of problem, a digital signature mechanism, discussed later in this chapter, can be used.

• *Replication*. An attacker might trap a packet and replay it. This kind of attack can be detected by using the sequence number for each packet.

• *Ping of death*. An attacker may send a *ping message*, which is large and therefore must be fragmented for transport. The receiver then starts to reassemble the fragments as the ping fragments arrive. The total packet length becomes too large and might cause a system crash.

• *Malicious misrouting of packets*. A hacker may attack a router and change its routing table, resulting in misrouting of data packets, causing a denial of service.

Denial-of-Service Attacks

A *denial-of-service attack* is a type of security breach that prohibits a user from accessing normally provided services. The denial of service does not result in information theft or any kind of information loss but can nonetheless be very dangerous, as it can cost the target person a large amount of time and money. Denial-of-service attacks affect the destination rather than a data packet or router.

Usually, a denial-of-service attack affects a specific network service, such as e-mail or DNS. For example, such an attack may overwhelm the DNS server in various ways and make it inoperable. One way of initiating this attack is by causing buffer overflow. Inserting an executable code inside memory can potentially cause a buffer overflow. Or, an adversary may use various tools to send large numbers of queries to a DNS server, which then is not able to provide services in a timely manner.

Denial-of-service attacks are easy to generate but difficult to detect. They take important servers out of action for few hours, thereby denying service to all users. There are yet a few other situations that can cause this kind of attack, such as UDP flood, a TCP flood and ICMP flood. In all these attacks, the hacker's main aim is to overwhelm victims and disrupt services provided to them.

Denial-of-service attacks are two types

1. *Single-source*. An attacker sends a large number of packets to a target system to overwhelm and disable it. These packets are designed such that their real sources cannot be identified.

2. *Distributed*. In this type of attack, a large number of hosts are used to flood unwanted traffic to a single target. The target cannot then be accessible to other users in the network, as it is processing the flood of traffic.

The flood may be either a UDP flood or a TCP SYN flood. UDP flooding is used against two target systems and can stop the services offered by either system. Hackers link the UDP character-generating services of a system to another one by sending UDP packets with spoofed return addresses. This may create an infinite looping between the two systems, leading to system uselessness.

Normally, a SYN packet is sent by a host to a user who intends to establish a connection. The user then sends back an acknowledgment. In the TCP SYN flood, a hacker sends a large number of SYN packets to a target user. Since the return addresses are spoofed, the target user queues up a SYN/ACK packet and never processes it. Therefore, the target system keeps on waiting. The result may be a hard disk crash or reboot.

SECURITY METHODS

Common solutions that can protect computer communication networks from attacks are classified as *cryptographic techniques* or *authentication techniques* (verification).

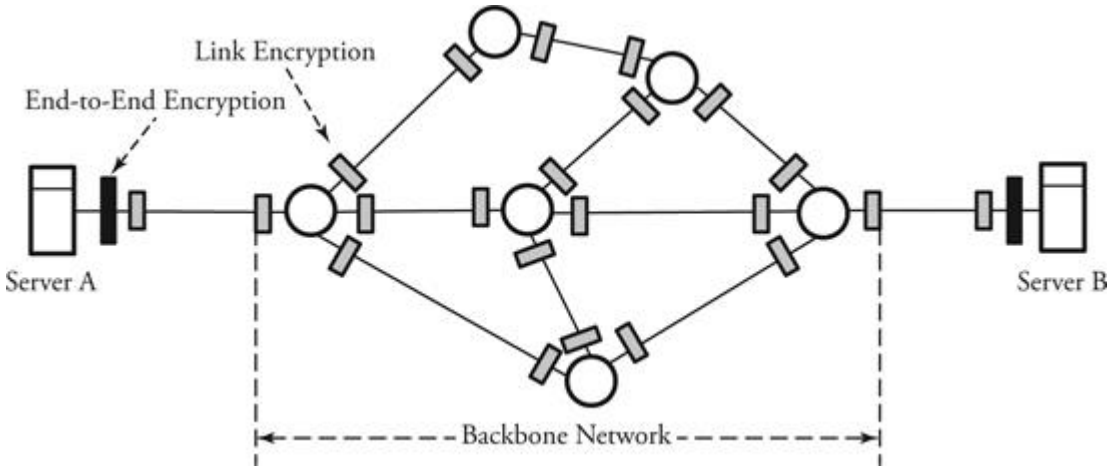
Cryptographic Techniques

Cryptography has a long and fascinating history. Centuries ago, cryptography was used as a tool to protect national secrets and strategies. Today, network engineers focus on *cryptography* methods for computer communication networks. Cryptography is the process of tranforming a piece of information or message shared by two parties into some sort of code. The message is scrambled before transmission so that it is undetectable by outside watchers. This kind of message needs to be decoded at the receiving end before any further processing.

The main tool that network security experts are using to encrypt a message M is a secret key K; the fundamental operation often used to encrypt a message is the Exclusive-OR ($\oplus$). Suppose that we have one bit, M, and a secret bit, K. A simple encryption is carried out using $M(\oplus)$ K. To decrypt this message, the second party—if he/she has the key, K—can easily detect M by performing the following:

In computer communication networks, data can travel between two users while it is encrypted. In Figure 10.2, two servers are exchanging data while two types of encryption devices are installed in their communication network. The first encryption device is *end-to-end encryption*, whereby secret coding is carried out at both end systems. In this figure, server A encodes its data, which can be decoded only at the other end server. The second phase of encryption is *link encryption*, which secures all the traffic passing over that link.

Figure 10.2 Overview of encryption points in a communication network



The two types of encryption techniques are *secret-key encryption* and *public-key encryption*. In a secret-key model, both sender and receiver conventionally use the same key for an encryption process. In a public-key model, a sender and a receiver each use a different key. The public-key system is more powerful than the secret-key system and provides better security and message privacy. But the biggest drawback of public-key encryption is speed. The public-key system is significantly more complex computationally and may not be practical in many cases. Hence, the public-key system is used only to establish a session to exchange a session key. Then, this session key is used in a secret-key system for encrypting messages for the duration of the session.

Authentication Techniques

Encryption methods offer the assurance of message confidentiality. However, a networking system must be able to verify the authenticity of the message and the sender of the message. These forms of security techniques in computer networks are known as *authentication techniques* and are categorized as *authentication with message digest* and *authentication with digital signature.* Message authentication protects a user in a network against data falsification and ensures data integrity. These methods do not necessarily use keys.

SECRET-KEY ENCRYPTION PROTOCOLS

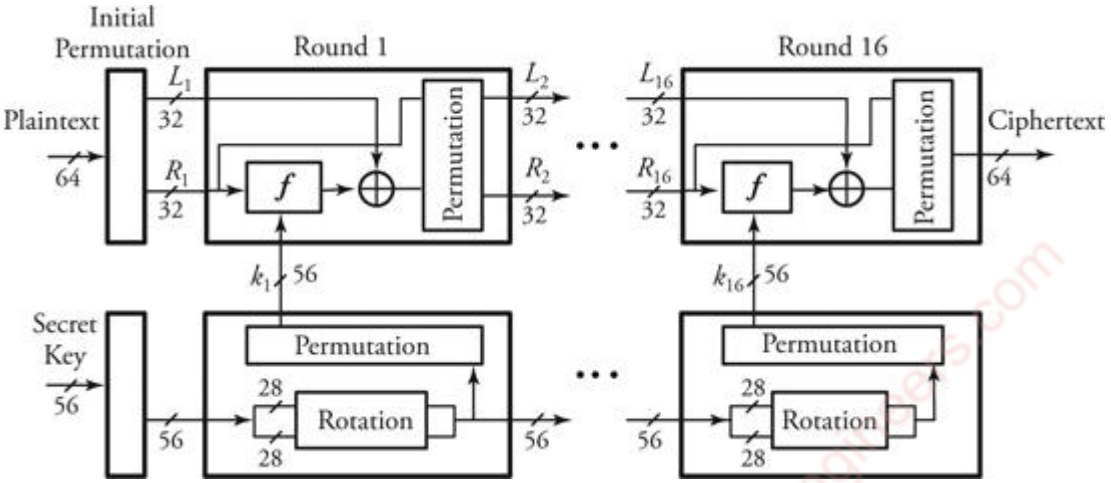
Secret-key encryption protocols, sometimes known as *symmetric encryption*, or *single-key encryption* protocols, are conventional encryption models. They typically consist of an encryption algorithm, a key, and a decryption algorithm. At the end point, the encrypted message is called *ciphertext*. Several standard mechanisms can be used to implement a secret-key encryption algorithm. Here, we focus on two protocols: *Data Encryption Standard* (DES) and *Advanced Encryption Standard* (AES).

In these algorithms, a shared secret key between a transmitter and a receiver is assigned at the transmitter and receiver points. The encryption algorithm produces a different key at any time for a specific transmission. Changing the key changes the output of the algorithm. At the receiving end, the encrypted information can be transformed back to the original data by using a decryption algorithm and the same key that was used for encryption. The security of conventional encryption depends on the secrecy of the key, not on the secrecy of the encryption algorithm. Consequently, the algorithm need not be kept secret; only the key has to be secret.

Data Encryption Standard (DES)

With the *Data Encryption Standard* (DES), plaintext messages are converted into 64-bit blocks, each encrypted using a key. The key length is 64 bits but contains only 56 usable bits; thus, the last bit of each 8 byte in the key is a parity bit for the corresponding byte. DES consists of 16 identical rounds of an operation, as shown in Figure 10.3. The details of the algorithm on each 64-bit block of message at each round *i* of operation are as follows.

Figure 10.3 The Data Encryption Standard (DES)



Begin DES Algorithm

1. Initialize. Before round 1 begins, all 64 bits of an incoming message and all 56 bits of the secret key are separately permuted (shuffled).

2. Each incoming 64-bit message is broken into two 32-bit halves denoted by L_i and R_i, respectively.

3. The 56 bits of the key are also broken into two 28-bit halves, and each half is rotated one or two bit positions, depending on the round.

4. All 56 bits of the key are permuted, producing version k_i of the key on round *i*.

5. In this step, is a logic Exclusive-OR, and the description of function *F*() appears next. Then, L_i and R_i are determined by

(10.2)

$$L_i = R_{i-1}$$

and

(10.3)

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i).$$

6. All 64 bits of a message are permuted.

The operation of function *F*() at any round *i* of DES is as follows.

1. Out of 52 bits of k_i, function *F*() chooses 48 bits.

2. The 32-bit R_{i-1} is expanded from 32 bits to 48 bits so that it can be combined with 48-bit k_i. The expansion of R_{i-1} is carried out by first breaking R_{i-1} into eight 4-bit chunks and then expanding each chunk by copying the leftmost bit and the rightmost bit from left and right adjacent chunks, respectively.

3. Function $F()$ also partitions the 48 bits of k_i into eight 6-bit chunks.

4. The corresponding eight chunks of R_{i-1} and eight chunks of k_i are combined as follows:

(10.4) $$R_{i-1} = R_{i-1} \oplus k_i.$$

At the receiver, the same steps and the same key are used to reverse the encryption. It is now apparent that the 56-bit key length may not be sufficient to provide full security. This argument is still controversial. Triple DES provides a solution for this controversy: three keys are used, for a total of 168 bits. It should also be mentioned that DES can be implemented more efficiently in hardware than in software.

Advanced Encryption Standard (AES)

The *Advanced Encryption Standard* (AES) protocol has a better security strength than DES. AES supports 128-bit symmetric block messages and uses 128-, 192-, or 256-bit keys. The number of rounds in AES is variable from 10 to 14 rounds, depending on the key and block sizes. Figure 10.4 illustrates the encryption overview of this protocol, using a 128-bit key. There are ten rounds of encryptions for the key size of 128 bits. All rounds are identical except for the last round, which has no mix-column stage.

Overview of Advanced Encryption Standard (AES) protocol

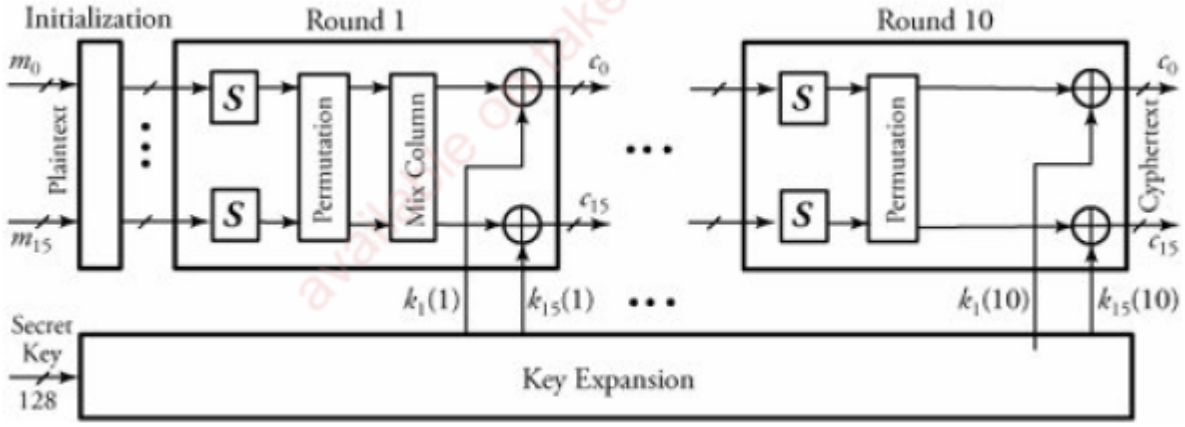


Figure 10.4. Overview of Advanced Encryption Standard (AES) protocol

A single block of 128-bit plaintext (16 bytes) as an input arrives from the left. The plaintext is formed as 16 bytes m_0 through m_{15} and is fed into round 1 after an initialization stage. In this round, substitute units—indicated by *S* in the figure—perform a byte-by-byte substitution of blocks. The ciphers, in the form of rows and columns, move through a *permutation stage* to shift rows to mix columns. At the end of this round, all 16 blocks of ciphers are Exclusive-ORed with the 16 bytes of round 1 key $k_0(1)$ through $k_{15}(1)$. The 128-bit key is expanded for ten rounds. The AES *decryption algorithm* is fairly simple and is basically the reverse of the encryption algorithm at each stage of a round. All stages of each round are reversible.

PUBLIC-KEY ENCRYPTION PROTOCOLS

The introduction of *public-key encryption* brought a revolution to the field of cryptography. Public-key cryptography provided a very clever method for key exchange. In the public-key encryption model, a sender/receiver pair uses different keys. This model is sometimes known as *asymmetric*, or *two-key*, *encryption*.

Public-key algorithm is based on mathematical functions rather than on substitution or permutation, although the security of any encryption scheme indeed depends on the length of the key and the computational work involved in breaking an encrypted message. Several public-key encryption protocols can be implemented. Among them, the following two protocols are the focus of our study:

- *Rivert, Shamir, and Aldeman* (RSA) protocol
- *Diffie-Hillman key-exchange* protocol.

In the public-key encryption methods, either of the two related keys can be used for encryption; the other one, for decryption. It is computationally infeasible to determine the decryption key given only the algorithm and the encryption key. Each system using this encryption method generates a pair of keys to be used for encryption and decryption of a message that it will receive. Each system publishes its encryption key by placing it in a public register or file and sorts out the key as a public one.

The companion key is kept private. If *A* wishes to send a message to *B*, *A* encrypts the message by using *B*'s public key. On receiving the message, *B* decrypts it with the private *B* key. No other recipients can decrypt the message, since only *B* knows its private key. This way, public-key encryption secures an incoming communication as long as a system controls its private key. However, public-key encryption has extra computational overhead and is more complex than the conventional one.

RSA Algorithm

Rivert, Shamir, and Aldeman developed the RSA public-key encryption and signature scheme. This was the first practical public-key encryption algorithm. RSA is based on the intractability of factoring large integers. Assume that a plaintext *m* must be encrypted to a ciphertext *c*. The RSA algorithm has three phases for this: *key generation*, *encryption*, and *decryption*.

Key Generation

In the RSA scheme, the key length is typically 512 bits, which requires an enormous computational power. A plaintext is encrypted in blocks, with each block having a binary value less than some number *n*. Encryption and decryption are done as follows, beginning with the generation of a public key and a private key.

Begin Key Generation Algorithm

1. Choose two roughly 256-bit prime numbers, *a* and *b*, and derive $n = ab$. (A number is prime if it has factors of 1 and itself.)

2. Find *x*. Select encryption key *x* such that *x* and $(a - 1)(b - 1)$ are relatively prime. (Two numbers are relatively prime if they have no common factor greater than 1.)

3. Find *y*. Calculate decryption key *y*:

(10.5) $$xy \bmod (a-1)(b-1) = 1.$$

4. At this point, *a* and *b* can be discarded.

5. The public key = {*x*, *n*}.

6. The private key = {*y*, *n*}.

In this algorithm, *x* and *n* are known to both sender and receiver, but only the receiver must know *y*. Also, *a* and *b* must be large and about the same size and both greater than 1,024 bits. The larger these two values, the more secure the encryption.

Encryption

Both sender and receiver must know the value of *n*. The sender knows the value of *x*, and only the receiver knows the value of *y*. Thus, this is a public-key encryption, with the public key {*x*, *n*} and the private key {*y*, *n*}. Given $m < n$, ciphertext *c* is constructed by

(10.6) $$c = m^x \bmod n.$$

Note here that if *a* and *b* are chosen to be on the order of 1,024 bits, $n \approx 2,048$. Thus, we are not able to encrypt a message longer than 256 characters.

Decryption

Given the ciphertext, *c*, the plaintext, *m*, is extracted by

(10.7) $$m = c^y \bmod n.$$

In reality, the calculations require a math library, as numbers are typically huge. One can see easily how Equations (10.6) and (10.7) work.

Example. For an RSA encryption of a 4-bit message of 1,000, or $m = 9$, we choose $a = 3$ and $b = 11$. Find the public and the private keys for this security action, and show the ciphertext.

Solution. Clearly, $n = ab = 33$. We select $x = 3$, which is relatively prime to $(a - 1)(b - 1) = 20$. Then, from $xy \bmod (a - 1)(b - 1) = 3y \bmod 20 = 1$, we can get $y = 7$. Consequently, the public key and the private key should be {3, 33} and {7, 33}, respectively. If we encrypt the message, we get $c = m^x \bmod n = 9^3 \bmod 33 = 3$. The decryption process is the reverse of this action, as $m = c^y \bmod n = 3^7 \bmod 33 = 9$.

Diffie-Hillman Key-Exchange Protocol

In the *Diffie-Hillman key-exchange* protocol, two end users can agree on a shared secret code without any information shared in advance. Thus, intruders would not be able to access the transmitted communication between the two users or discover the shared secret code. This protocol is normally used for *virtual private networks* (VPNs), explained in Chapter 16. The essence of this protocol for two users, 1 and 2, is as follows. Suppose that user 1 selects a prime *a*, a random integer number x_1, and a generator *g* and creates y_1 {1, 2, ..., $a - 1$} such that

(10.8) $$y_1 = g^{x_1} \bmod a.$$

In practice, the two end users agree on a and g ahead of time. User 2 performs the same function and creates y_2:

(10.9) $$y_2 = g^{x_2} \bmod a.$$

User 1 then sends y_1 to user 2. Now, user 1 forms its key, k_1, using the information its partner sent as

(10.10) $$k_1 = y_2^{x_1} \bmod a,$$

and user 2 forms its key, k_2, using the information its partner sent it as

(10.11) $$k_2 = y_1^{x_2} \bmod a.$$

It can easily be proved that the two Keys k_1 and k_2 are equal. Therefore, the two users can now encrypt their messages, each using its own key created by the other one's information.

AUTHENTICATION

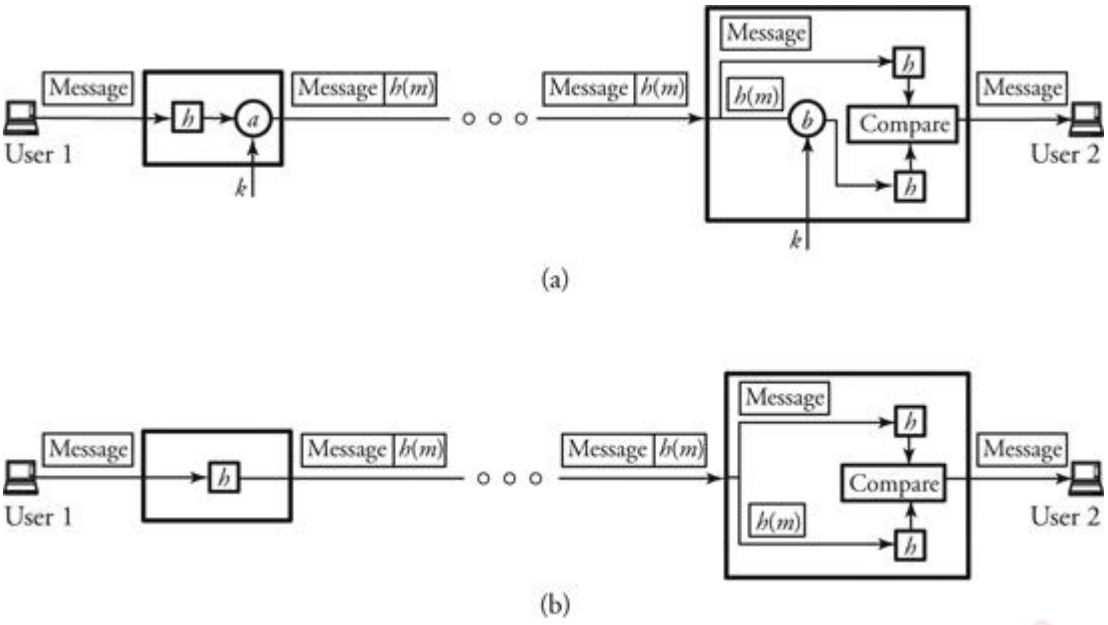
Authentication techniques are used to verify identity. Message authentication verifies the authenticity of both the message content and the message sender. Message content is authenticated through implementation of a hash function and encryption of the resulting message digest. The sender's authenticity can be implemented by use of a digital signature.

A common technique for authenticating a message is to implement a *hash function*, which is used to produce a "fingerprint" of a message. The hash value is added at the end of message before transmission. The receiver recomputes the hash value from the received message and compares it to the received hash value. If the two hash values are the same, the message was not altered during transmission. Once a hash function is applied on a message, m, the result is known as a message digest, or $h(m)$. The hash function has the following properties.

- Unlike the encryption algorithm, the authentication algorithm is not required to be reversible.
- Given a message digest $h(m)$, it is computationally infeasible to find m.
- It is computationally infeasible to find two different messages m_1 and m_2 such that $h(m_1) = h(m_2)$.

Message authentication can be implemented by two methods. In the first method, as shown in Figure 10.5 (a), a hash function is applied on a message, and then a process of encryption is implemented. Thus, a message digest can also be encrypted in this method. At this stage, the encryption can be a public key or a secret key. The authenticity of a message in this method is assured only if the sender and the receiver share the encryption key. At the receiver site, the receiving user 2 has to decrypt the received message digest and compare it with the one made locally at its site for any judgments on the integrity of the message.

Figure 10.5 Message authentication: (a) combined with encryption; (b) use of the hash function



In the second method, as shown in Figure 10.5 (b), no encryption is involved in the process of message authentication. This method assumes that the two parties share a secret key. Hence, at the receiving site, the comparison is made between the received $h(m)$ and the message digest made locally from the received message. This technique is more popular in the security infrastructure of the Internet Protocol. Among the message authentication protocols are the MD5 *hash algorithm* and the *Secure Hash Algorithm* (SHA). SHA is the focus of our discussion.

Secure Hash Algorithm (SHA)

The *Secure Hash Algorithm* (SHA) was proposed as part of the digital signature standard. SHA-1, the first version of this standard, takes messages with a maximum length of 2^{24} and produces a 160-bit digest. With this algorithm, SHA-1 uses five registers, R_1 through R_5, to maintain a "state" of 20 bytes.

The first step is to pad a message m with length l_m. The message length is forced to $l_m = 448 \bmod 512$. In other words, the length of the padded message becomes 64 bits less than the multiple of 512 bits. The number of padding bits can be as low as 1 bit and as high as 512 bits. The padding includes a 1 bit and as many 0 bits as required. Therefore, the least-significant 64 bits of the message length are appended to convert the padded message to a word with a multiple of 512 bits.

After padding, the second step is to expand each block of 512-bit (16 32 bits) words $\{m_0, m_1, ..., m_{15}\}$ to words of 80 32 bits using:

(10.12) $$w_i = m_i \text{ for } 0 \le i \le 15$$

And

(10.13) $$w_i = w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} \hookleftarrow 1 \text{ for } 16 \le i \le 79,$$

Where $\hookleftarrow j$ means left rotation by j bits. This way, bits are shifted several times if the incoming block is mixed with the state. Next, bits from each block of w_i are mixed into the state in four steps, each maintaining 20 rounds. For any values of a, b, and c, and bit number i, we define a function $F_i(a, b, c)$ as follows:

(10.14)

$$F_i(a,b,c) = \begin{cases} (a \cap b) \cup (\bar{a} \cap c) & 0 \le i \le 19 \\ a \oplus b \oplus c & 20 \le i \le 39 \\ (a \cap b) \cup (a \cap c) \cup (b \cap c) & 40 \le i \le 59 \\ a \oplus b \oplus c & 60 \le i \le 79 \end{cases}.$$

Then, the 80 steps (i = 0, 1, 2, ..., 79) of the four rounds are described as follows:

(10.15)

$$\delta = (R_1 \hookleftarrow 5) + F_i(R_2, R_3, R_4) + R_5 + w_i + C_i$$

(10.16) $R_5 = R_4$

(10.17) $R_4 = R_3$

(10.18) $R_3 = R_2 \hookleftarrow 30$

(10.19) $R_2 = R_1$

(10.20) $R_1 = \delta,$

where C_i is a constant value specified by the standard for round i. The message digest is produced by concatenation of the values in R_1 through R_5.

Authentication and Digital Signature

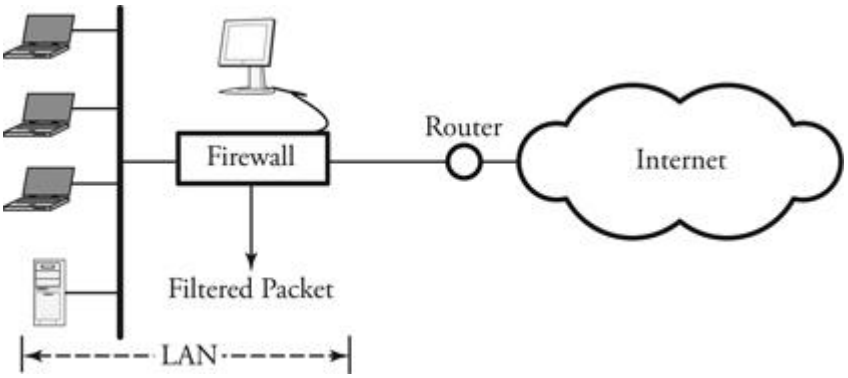
A digital signature is one of the most important required security measures. Much like a person's signature on a document, a digital signature on a message is required for the authentication and identification of the right sender. The digital signature is supposed to be unique to an individual and serves as a means of identifying the sender. An electronic signature is not as easy as it was with the paper-based system. The digital signature requires a great deal of study, research, and skill. Even if these requirements are met, there is no guarantee that all the security requirements have been met.

The technical method of providing a sender's authentication is performed through cryptography. Many cryptographic mechanisms have been developed. Among them, the RSA algorithm implements both encryption and digital signature. When RSA is applied, the message is encrypted with the sender's private key. Thus, the entire encrypted message serves as a digital signature. This means that at the receiving end, the receiver can decrypt it, using the public key. This authenticates that the packet comes from the right user .

FIREWALLS

As the name suggests, a *firewall* protects data from the outside world. A firewall can be a software program or a hardware device. A firewall is a popular security mechanism for networks. A firewall is a simple router implemented with a special program. This unit is placed between hosts of a certain network and the outside world, as shown in Figure 10.8, and the rest of the network. The security issues faced by a smaller network like the one used at home are similar to larger networks. A firewall is used to protect the network from unwanted Web sites and potential hackers.

Figure 10.8 A simple configuration of a secured network using a firewall



A firewall is placed on the link between a network router and the Internet or between a user and a router. The objective of such a configuration is to monitor and filter packets coming from unknown sources. Consequently, hackers do not have access to penetrate through a system if a firewall protects the system. For a large company with many small networks, the firewall is placed on every connection attached to the Internet. Companies can set rules about how their networks or particular systems need to work in order to maintain security. Companies can also set rules on how a system can connect to Web sites. These precautionary rules are followed in order to attain the advantage of having a firewall. Hence, the firewall can control how a network works with an Internet connection. A firewall can also be used to control data traffic.

Software firewall programs can be installed in home computers by using an Internet connection with these so-called gateways, the computer with such a software can access Web servers only through this software firewall. But hardware firewalls are more secure than software firewalls. Moreover, hardware firewalls are not expensive. Some firewalls also offer virus protection. The biggest security advantage of installing a firewall in a business network is to protect from any outsider logging on to the network under protection. Firewalls are preferred for use in almost all network security infrastructures, as they allow the implementation of a security policy in one centralized place rather than end to end. Sometimes, a firewall is put where a large amount of data flow is normally expected.

A firewall controls the flow of traffic by one of the following three methods. The first method is *packet filtering*. Apart from forwarding packets between networks, a firewall filters those packets that pass through. If packets can get through the filter, they reach their destinations; otherwise, they are discarded. A firewall can be programmed to throw away certain packets addressed to a particular IP host or TCP port number. This condition is especially useful if a particular host does not want to respond to any access from an external source.

The second method is that a firewall filters packets based on the source IP address. This filtering is helpful when a host has to be protected from any unwanted external packets. The third method, denial of service, was explained earlier. This method controls the number of packets entering a network.

Internet routing is based on a distributed system of many routers, switches, and protocols. These protocols have a number of points of vulnerabilities that can be exploited to cause such problems as misdelivery or nondelivery of user traffic, misuse of network resources, network congestion and packet delays, and the violation of local routing policies.