CHAPTER 29

Cryptography

We begin our discussion of network security with an introduction to cryptography and a discussion of the methods used in security management. The science of cryptography is very complex; there are entire books devoted to the subject. A cryptography expert needs to be knowledgeable in areas such as mathematics, electronics, and programming. In this chapter, we consider the concepts needed to understand the security issues discussed in Chapter 30 and network security discussed in Chapter 31.

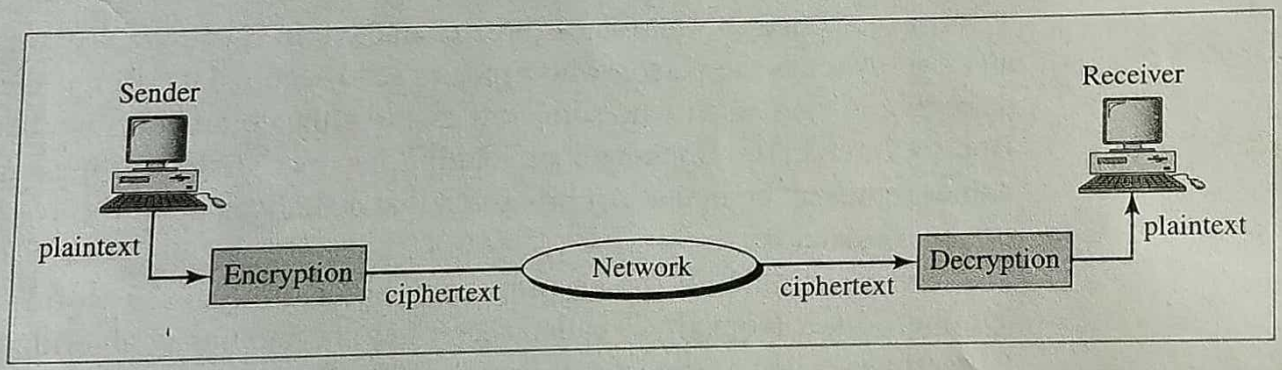
We focus on symmetric-key cryptography, which is presently more common than public-key cryptography. Symmetric-key cryptography is less math-based than public-key cryptography, which has its origins in number theory.

Cryptography and its applications to the Internet are a relatively new field whose importance increases with every new attack on the Internet.

29.1 INTRODUCTION

The word **cryptography** in Greek means "secret writing." However, the term today refers to the science and art of transforming messages to make them secure and immune to attacks. Figure 29.1 shows the components involved in cryptography.

Figure 29.1 *Cryptography components*



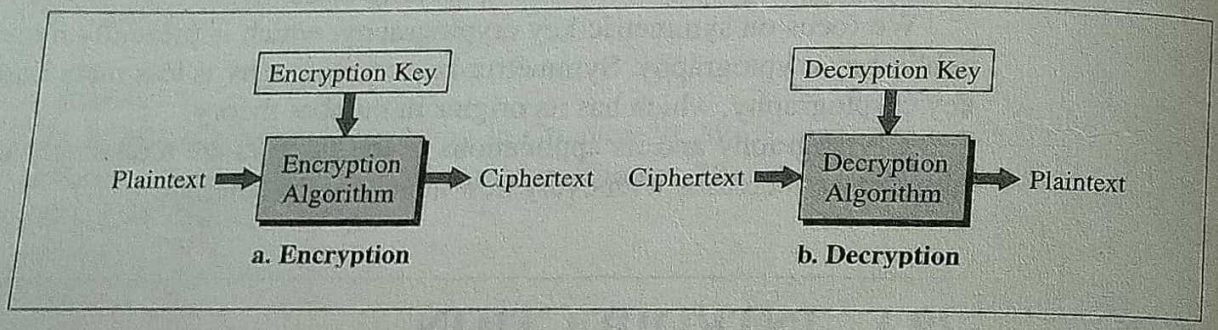
The original message, before being transformed, is called **plaintext.** After the message is transformed, it is called **ciphertext.** An **encryption** algorithm transforms the plaintext to ciphertext; a **decryption** algorithm transforms the ciphertext back to

plaintext. The sender uses an encryption algorithm, and the receiver uses a decryption algorithm.

Throughout this chapter and Chapters 30 and 31, we discuss encryption and decryption algorithms. We refer to them as **ciphers.** The term *cipher* is also used to refer to different categories of algorithms in cryptography.

This is not to say that every sender-receiver pair needs its very own unique cipher for a secure communication. Instead, through the use of public ciphers with secret keys, one cipher can serve millions of communicating pairs. A **key** is a number (value) that the cipher, as an algorithm, operates on. To encrypt a message, we need an encryption algorithm, an encryption key, and the plaintext. These create the ciphertext. To decrypt a message, we need a decryption algorithm, a decryption key, and the ciphertext. These reveal the original plaintext. Figure 29.2 shows the idea.

Figure 29.2 *Encryption and decryption*



The encryption and decryption algorithms are public; anyone can access them. The keys are secret; they need to be protected.

In cryptography, the encryption/decryption algorithms are public; the keys are secret.

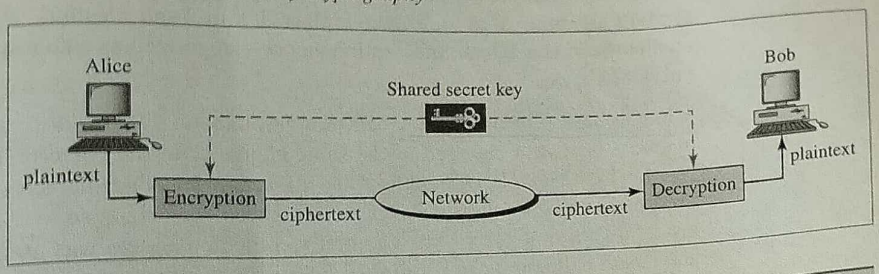
It is customary to introduce three characters in cryptography; we use Alice, Bob, and Eve. Alice is the person who needs to send secure data. Bob is the recipient of the data. Eve is the person who somehow disturbs the communication between Alice and Bob by intercepting messages or sending her own disguised messages. These three names represent computers or processes that actually send or receive data, or intercept or change data.

We can divide all the cryptography algorithms in the world into two groups: symmetric-key (sometimes called secret-key) cryptography algorithms and public-key (sometimes called asymmetric) cryptography algorithms.

29.2 SYMMETRIC-KEY CRYPTOGRAPHY

In symmetric-key cryptography, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data (see Fig. 29.3).

Figure 29.3 *Symmetric-key cryptography*



In symmetric-key cryptography, the same key is used by the sender (for encryption) and the receiver (for decryption). The key is shared.

In symmetric-key cryptography, the algorithm used for decryption is the inverse of the algorithm used for encryption. This means that if the encryption algorithm uses a combination of addition and multiplication, the decryption algorithm uses a combination of division and subtraction.

Note that the symmetric-key cryptography algorithms are so named because the same key can be used in both directions.

In symmetric-key cryptography, the same key is used in both directions.

Symmetric-key algorithms are efficient; it takes less time to encrypt a message using a symmetric-key algorithm than it takes to encrypt using a public-key algorithm. The reason is that the key is usually smaller. For this reason, symmetric-key algorithms are used to encrypt and decrypt long messages.

Symmetric-key cryptography is often used for long messages.

A symmetric-key algorithm has two major disadvantages. Each pair of users must have a unique symmetric key. This means that if N people in the world want to use this method, there needs to be $N(N-1)/2$ symmetric keys. For example, for 1 million people to communicate, 500 billion symmetric keys are needed. The distribution of the keys between two parties can be difficult. We will see how we can solve this problem in Chapter 30.

Traditional Ciphers

In the earliest and simplest ciphers, a character was the unit of data to be encrypted. These traditional ciphers involved either substitution or transposition.

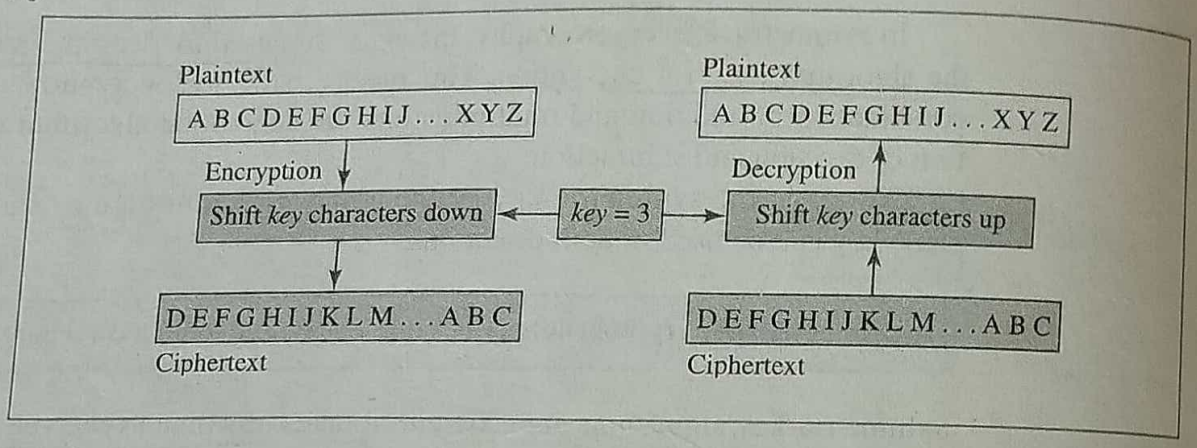
Substitution Cipher

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

another. For example, we can replace character A with D and character T with Z. If the symbols are digits (0 to 9), we can replace 3 with 7 and 2 with 6. We will concentrate on alphabetic characters. Substitution can be categorized as either **monoalphabetic** or **polyalphabetic.**

Monoalphabetic Substitution In monoalphabetic substitution, a character in the plaintext is always changed to the same character in the ciphertext regardless of its position in the text. For example, if the algorithm says that character A in the plaintext must be changed to character D, every character A is changed to character D, regardless of its position in the text. The first recorded ciphertext was used by Julius Caesar and is still called the *Caesar cipher*. The cipher shifts each character down by three. Figure 29.4 shows idea of the Caesar cipher.

Figure 29.4 *Caesar cipher*



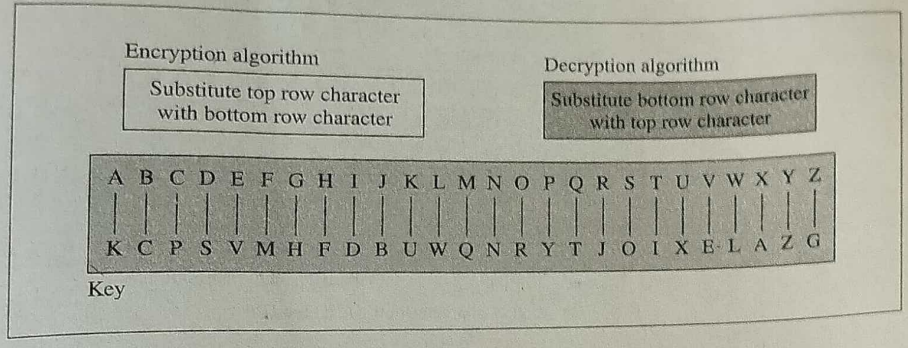
Before we go further, let us analyze the Caesar cipher which has an encryption algorithm, a decryption algorithm, and a symmetric key. As the figure shows, the encryption algorithm is "shift *key* characters down." The decryption algorithm is "shift *key* characters up." The key is 3. Note that the encryption and decryption algorithms are the inverses of each other; the key is the same in encryption and decryption.

We can think of monoalphabetic substitution in another way. We can assign numbers to the alphabet characters (A = 0, B = 1, C = 3, . . . , Z = 25). We can think of the encryption algorithm as simply "add the key to the plaintext number to get the ciphertext number." Decryption is the same, but we replace *add* with *subtract* and switch *plaintext* with *ciphertext*. Of course adding and subtracting are modulo 26, which means that 24 + 3 is 1, not 27; Y (24) is substituted with B (1).

In monoalphabetic substitution, the relationship between a character in the plaintext and a character in the ciphertext is always one-to-one. We can have many other encryption/decryption algorithms with other keys. We could change character A to J (shift of 9) and change character P to M (shift of –3). Figure 29.5 shows another example of monoalphabetic substitution. In this cipher, the two algorithms are still the inverse of each other. The key is the two rows as shown in the figure. Note that we still have a monoalphabetic substitution because the one-to-one relation is preserved.

Monoalphabetic substitution is very simple, but the code can be attacked easily. The reason is that the method cannot hide the natural frequencies of characters in the

Figure 29.5 *Example of monoalphabetic substitution*



In monoalphabetic substitution, the relationship between a character in the plaintext to the character in the ciphertext is always one-to one.

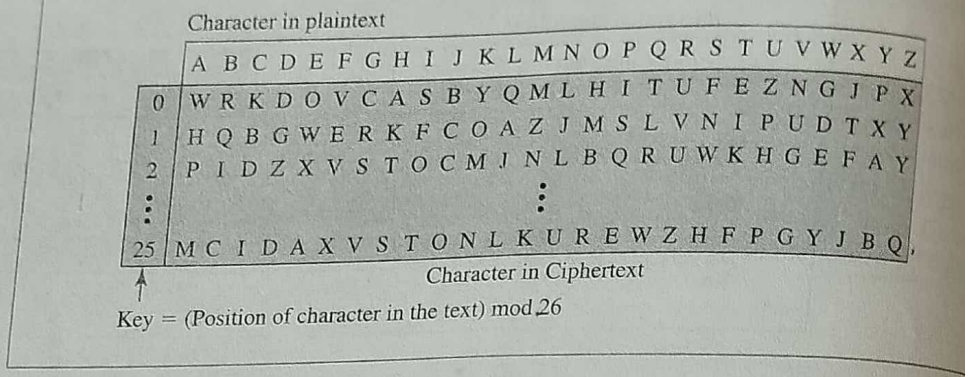
language being used. For example, in English, the most frequently used characters are E, T, O, and A. An attacker can easily break the code by finding which character is used the most and replace that one with the letter E. It can then find the next most frequent and replace it with T, and so on.

Polyalphabetic Substitution In *polyalphabetic substitution*, each occurrence of a character can have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many. Character A can be changed to D in the beginning of the text, but it could be changed to N at the middle. There are many interesting polyalphabetic substitution ciphers. We discuss a very simple one. It is obvious that if the relationship between plaintext characters and ciphertext characters is one-to-many, the key must tell us which of the many possible characters can be chosen for encryption. Let us define our key as "take the position of the character in the text, divide the number by 10, and let the remainder be the shift value." With this scenario, the character at position 1 will be shifted one character, the character at position 2 will be shifted two characters, and the character in position 14 will be shifted four characters (14 mod 10 is 4).

An example of polyalphabetic substitution is the **Vigenere cipher.** In one version of this cipher, the character in the ciphertext is chosen from a two-dimensional table (26×26), in which each row is a permutation of 26 characters (A to Z). To change a character, the algorithm finds the character to be encrypted in the first row. It finds the position of the character in the text (mod 26) and uses it as the row number. The algorithm then replaces the character with the character found in the table. Figure 29.6 shows only some of the rows and columns. According to this table, A is encrypted as W if it is in position 0 and as M if it is in position 25.

A ciphertext created by polyalphabetic substitution is harder to attack successfully than a ciphertext created by monoalphabetic substitution. A simple observation of the frequencies does not help. As a matter of fact, a good polyalphabetic substitution may smooth out the frequencies; each character in the ciphertext may occur almost the same

Figure 29.6 *Vigenere cipher*



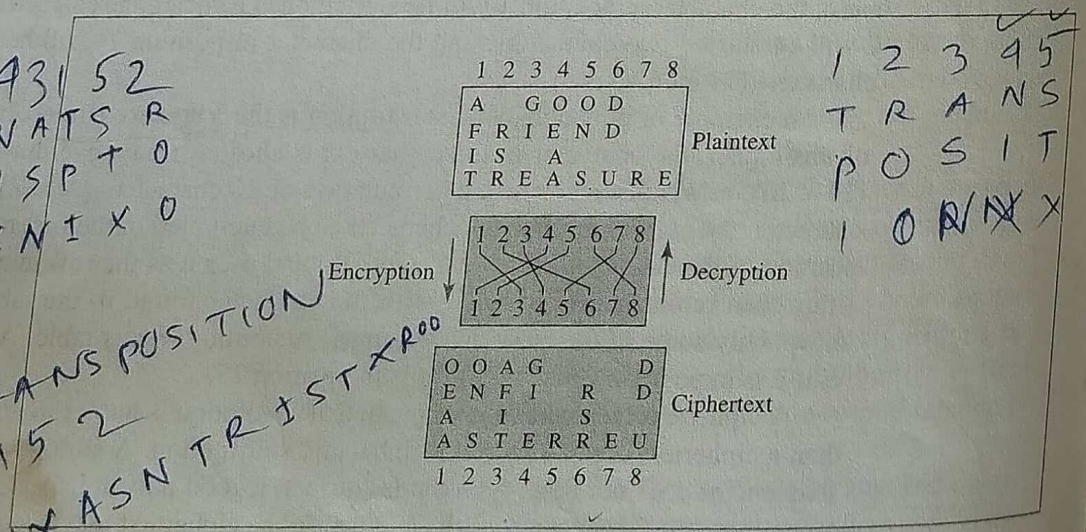
In polyalphabetic substitution, the relationship between a character in the plaintext and a character in the ciphertext is one-to-many.

number of times. However, attacking the code is not difficult; although the encryption changes the frequency of the characters, the character relationships are still preserved. A good trial-and-error attack can break the code. But we leave this as an exercise for students.

Transpositional Cipher

In a **transpositional cipher,** the characters retain their plaintext form but change their positions to create the ciphertext. The text is organized into a two-dimensional table, and the columns are interchanged according to a key. For example, we can organize the plaintext into an 8-column table and then reorganize the columns according to a key that indicates the interchange rule. Figure 29.7 shows an example

Figure 29.7 *Transpositional cipher*

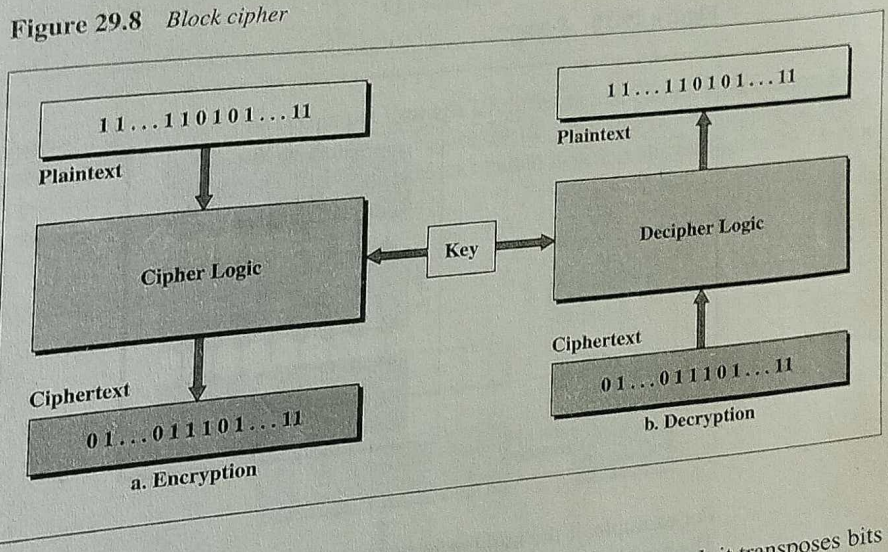


of transpositional cryptography. The key defines which columns should be swapped. As you have guessed, transpositional cryptography is not very secure either. The character frequencies are preserved, and the attacker can find the plaintext through trial and error. This method can be combined with other methods to provide more sophisticated ciphers.

Block Cipher

Traditional ciphers used a character or symbol as the unit of encryption/decryption. Modern ciphers, on the other hand, use a block of bits as the unit of encryption/decryption. Figure 29.8 shows the concept of the **block cipher;** the plaintext and ciphertext are blocks of bits.

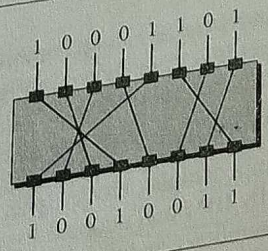
Figure 29.8 *Block cipher*



P-box

A **P-box** (P for permutation) performs a transposition at the bit level; it transposes bits as shown in Figure 29.9. It can be implemented in software or hardware, but hardware

Figure 29.9 *P-box*

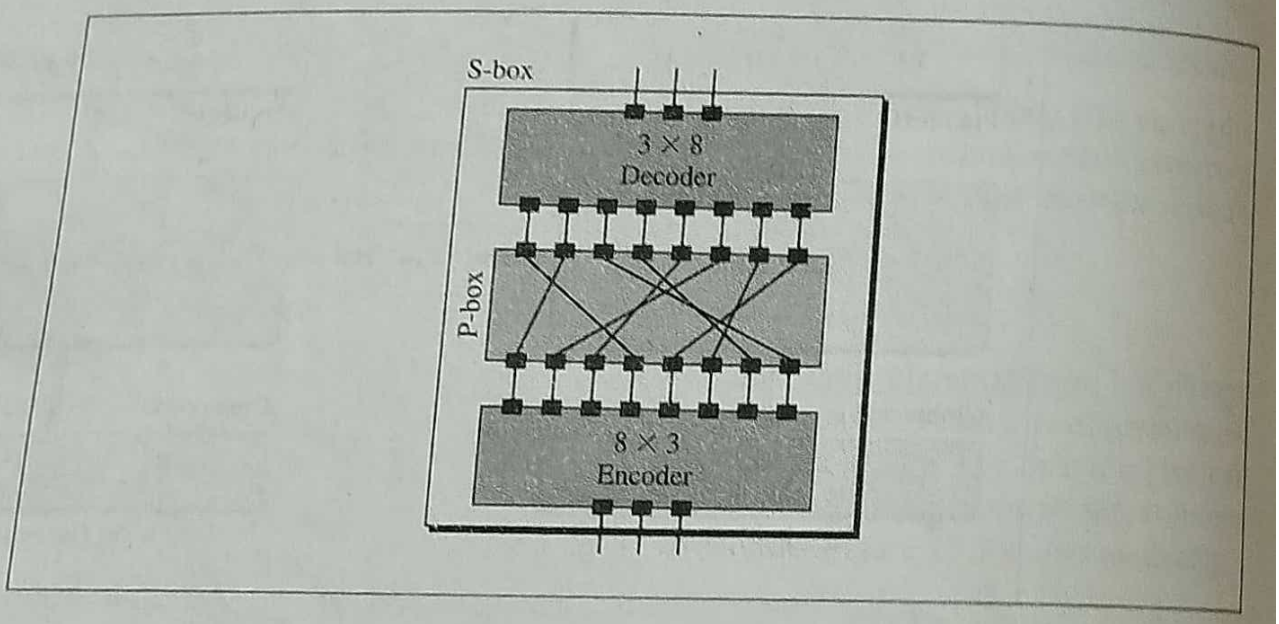


is faster. The key and the encryption/decryption algorithm are normally embedded in the hardware. Note that both the plaintext and ciphertext have the same number of 1s and 0s.

S-box

An **S-box** (S for substitution) performs a substitution at the bit level; it transposes permuted bits as shown in Figure 29.10. The S-box substitutes one decimal digit with another. The S-box normally has three components: an encoder, a decoder, and a P-box. The decoder changes an input of n bits to an output of 2^n bits. This output has one single 1 (the rest are 0s) located at a position determined by the input. The P-box permutes the output of decoder, and the encoder changes the output of the P-box back to a binary number in the same way as the decoder, but inversely.

Figure 29.10 *S-box*



For example, if the number in the figure is 2 (010), the decoder changes it to 00000100. The position of the 1 bit is, counting from the right with the leftmost bit at position 0, at position 2. After the P-box transposition, in this configuration, we have 01000000. The 1 bit is at position 6. Therefore, the encoder encodes this as binary 110. The 2 has been changed to decimal digit 6.

Product Block

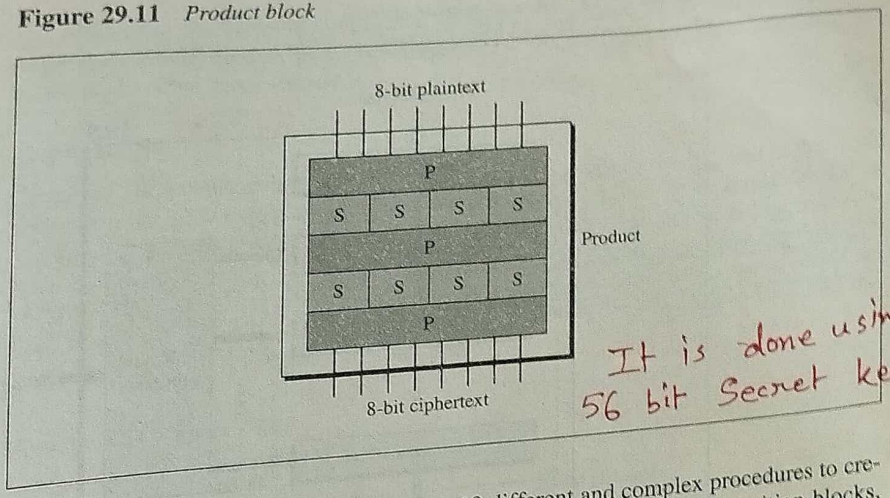
The P-boxes and S-boxes can be combined to get a more complex cipher block. This is called a **product block,** as shown in Figure 29.11.

Data Encryption Standard (DES)

Private Key Encryption. | Symmetric

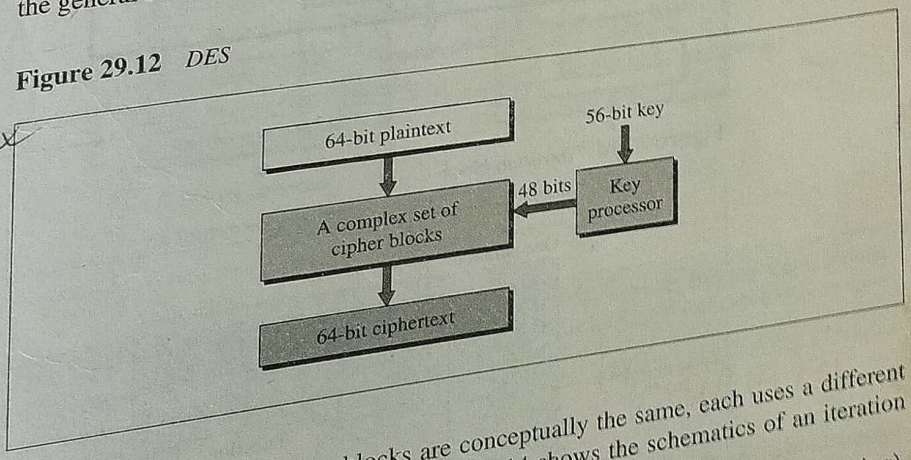
One example of a complex block cipher is the **Data Encryption Standard (DES).** DES was designed by IBM and adopted by the U.S. government as the standard encryption method for nonmilitary and nonclassified use. The algorithm encrypts a 64-bit plaintext

Figure 29.11 *Product block*



using a 56-bit key. The text is put through 19 different and complex procedures to create a 64-bit ciphertext, as shown in Figure 29.12. DES has two transposition blocks, one swapping block, and 16 complex blocks called iteration blocks. Figure 29.13 shows the general scheme.

Figure 29.12 *DES*



Although the 16 iteration blocks are conceptually the same, each uses a different key derived from the original key. Figure 29.14 shows the schematics of an iteration block.

In each block, the previous right 32 bits become the next left 32 bits (swapping). The next right 32 bits, however, come from first applying an operation (a function) on the previous right 32 bits and then XORing the result with the left 32 bits.

Note that the whole DES cipher block is a substitution block that changes a 64-bit plaintext to a 64-bit ciphertext. In other words, instead of substituting one character at a time, it substitutes 8 characters (bytes) at a time, using complex encryption and decryption algorithms.

Figure 29.13 *General scheme of DES*

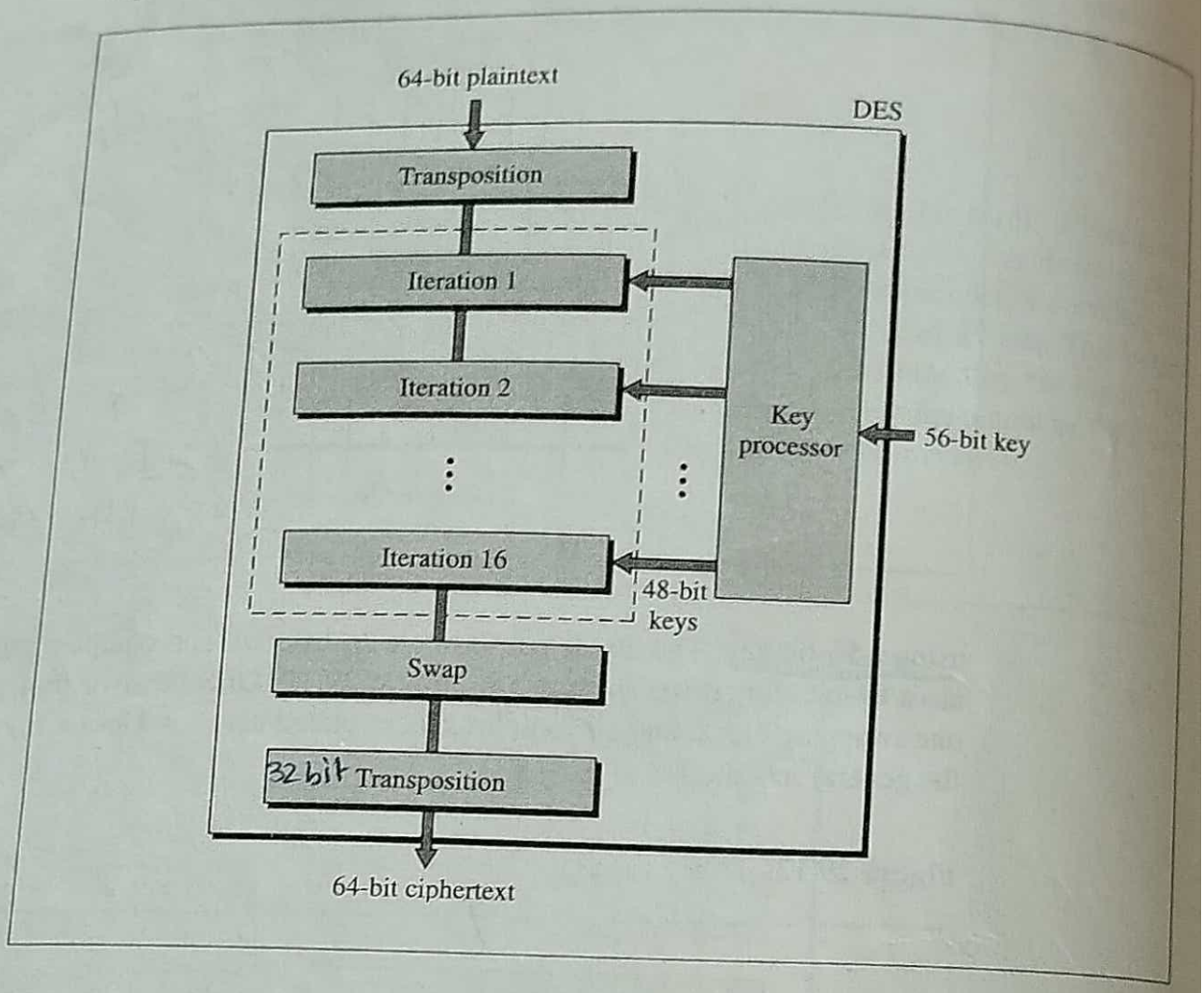
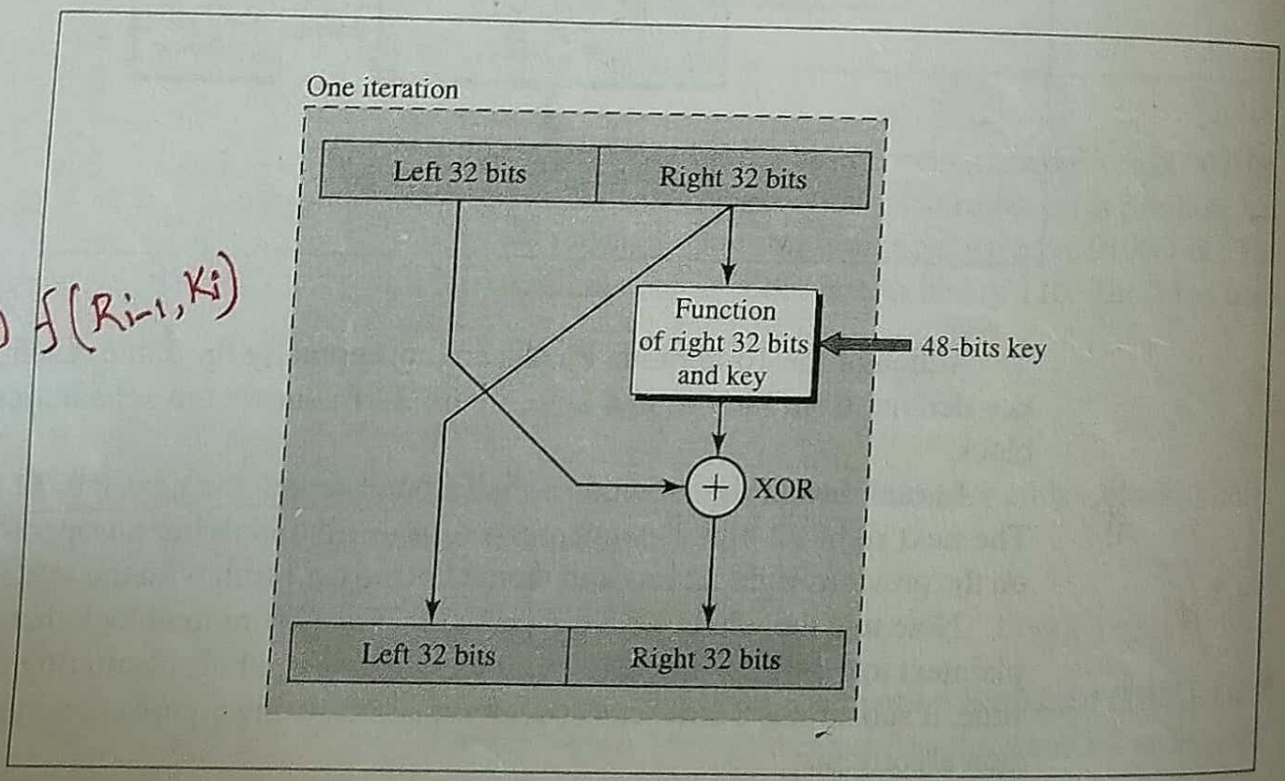


Figure 29.14 *Iteration block*



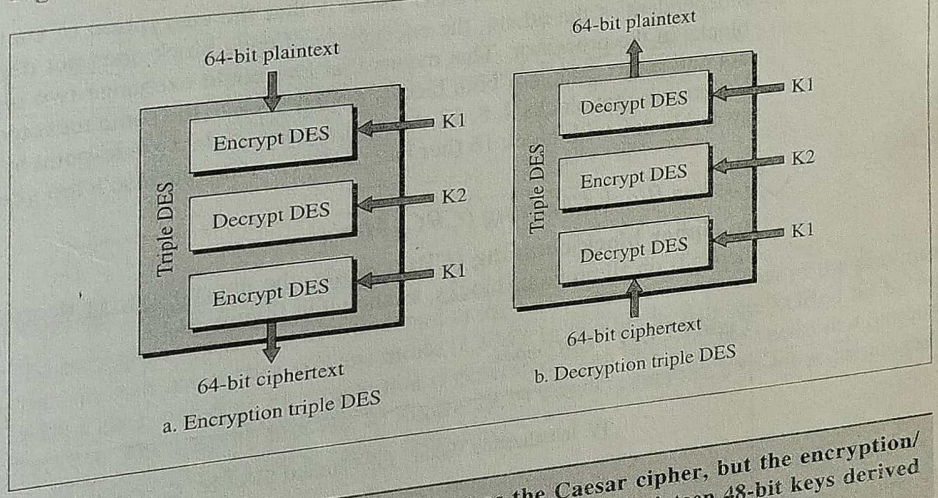
$L_{i-1} \oplus f(R_{i-1}, K_i)$

DES takes the data and chops them into 8-byte segments. However, the encryption and the key are the same for each segment. So if the data are four equal segments, the result is also four equal segments.

Triple DES

Critics of DES contend that the key is too short. To lengthen the key and at the same time keep the new block compatible with that of the original DES, **triple DES** was designed. This uses three DES blocks and two 56-bit keys, as shown in Figure 29.15. Note that the encrypting block uses an encryption-decryption-encryption combination of DESs, while the decryption block uses a decryption-encryption-decryption combination. It was designed this way to provide compatibility between triple DES and the original DES when K1 and K2 are the same.

Figure 29.15 *Triple DES*



The DES cipher uses the same concept as the Caesar cipher, but the encryption/decryption algorithm is much more complex due to the sixteen 48-bit keys derived from a 56-bit key.

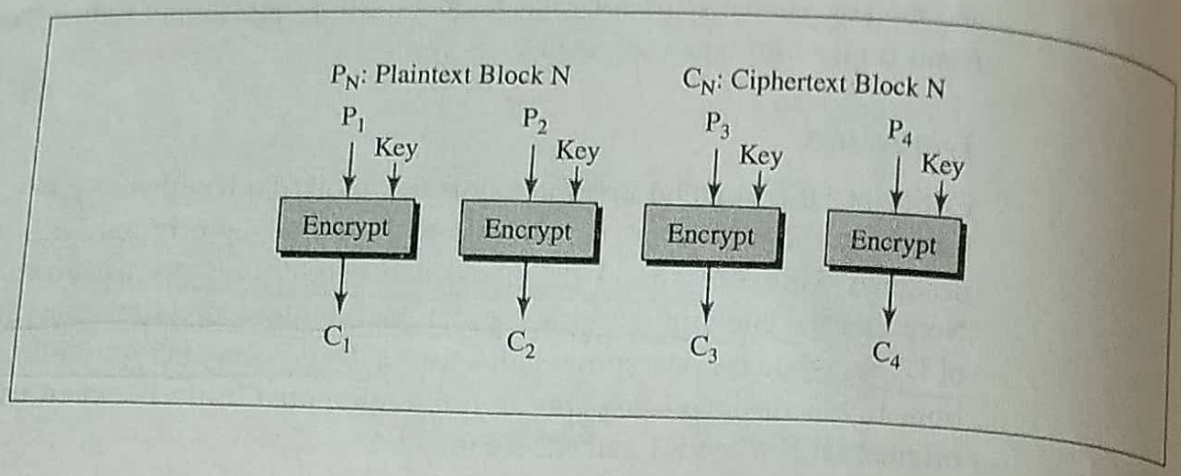
Operation Modes

DES and triple DES are actually long substitution ciphers that operate on eight-character segments (sometimes called long characters). Can we encrypt and decrypt longer messages (1000 characters, e.g.)? Several modes have been defined, and we briefly describe the four most common.

Electronic Code Block (ECB) Mode

In **electronic code block (ECB) mode,** we divide the long message into 64-bit blocks and encrypt each block separately, as shown in Figure 29.16. The encryption of each

Figure 29.16 *ECB mode*



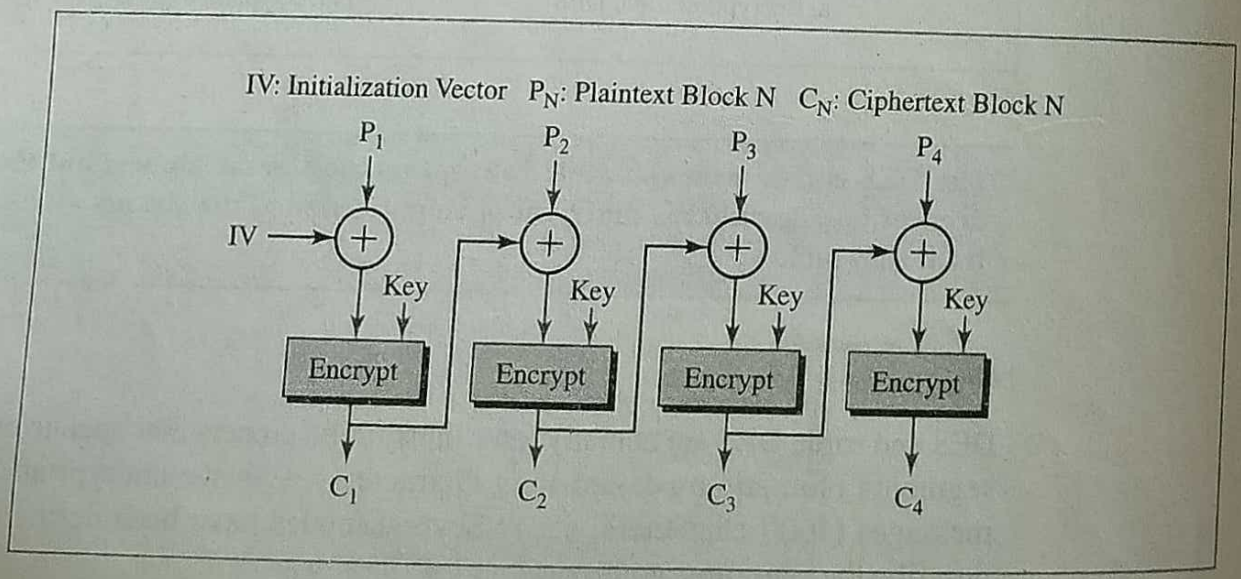
block is independent of the other blocks in ECB mode. Although the figure shows only four blocks, ECB is designed to handle many more.

The problem with the ECB mode is that the encryption of each 8-byte block is independent of the others; the encryption of each block does not depend on the other blocks in the processor. This means that Eve could exchange two blocks; Bob would not notice this change if both blocks were related to the same message. For example, if Eve knows that blocks 4, 8, 12, 16, . . . , are the student grade-point averages, she could swap block 8 with block 16 (her bad grade for someone else's top grade).

Cipher Block Chaining (CBC) Mode

In **cipher block chaining (CBC) mode,** the encryption (or decryption) of a block depends on all previous blocks, as shown in Figure 29.17.

Figure 29.17 *CBC mode*



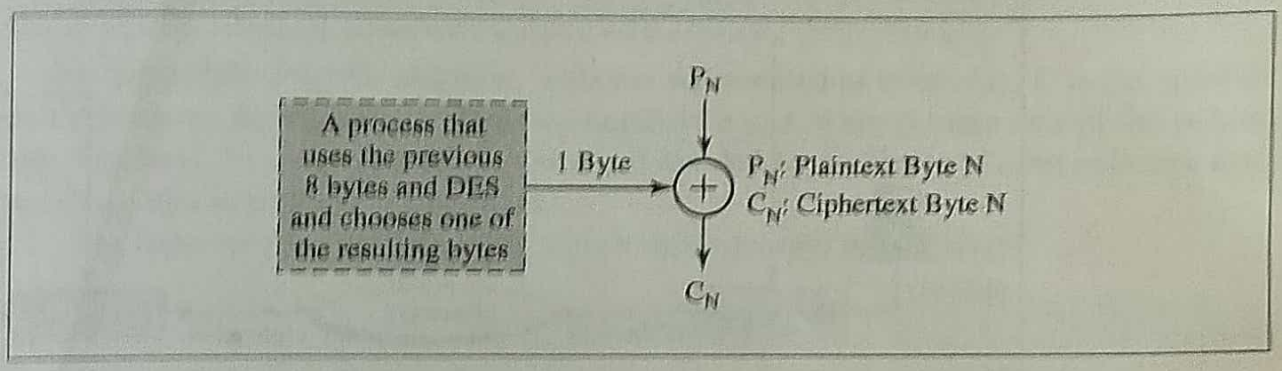
For example, to encrypt the second plaintext block (P_2), we first XOR it with the first ciphertext block (C_1) and then pass it through the encryption process. In this way, C_2 depends on C_1. If someone exchanges C_1 with C_3, for example, C_2 will

not decrypt correctly; it will create garbage. The situation for the first block is different because there is no C_0. Instead, a 64-bit random number, called the *initialization vector* (IV), is used. The IV is sent with the data so that the receiver can use it in decryption.

Cipher Feedback Mode (CFM)

Cipher feedback mode (CFM) was created for those situations in which we need to send or receive data 1 byte at a time, but still want to use DES (or triple DES). One solution is to make a 1-byte C_N dependent on a 1-byte P_N and another byte, which depends on 8 previous bytes itself, as shown in Figure 29.18.

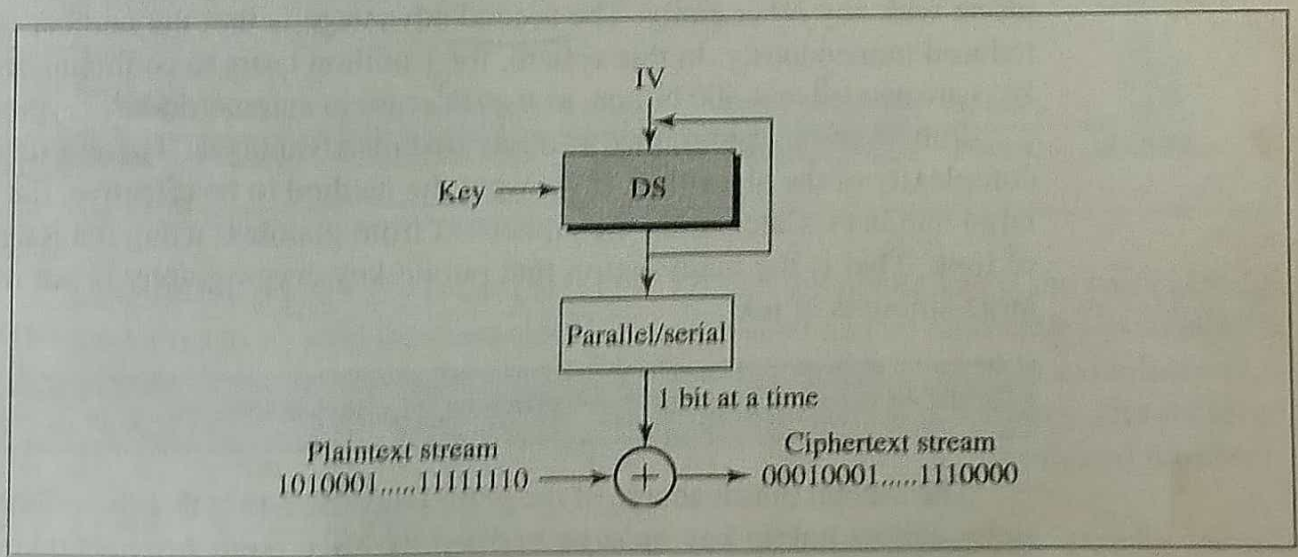
Figure 29.18 *CFM*



Cipher Stream Mode (CSM)

To encrypt/decrypt 1 bit at a time and at the same time be independent of the previous bits, we can use **cipher stream mode (CSM).** In this mode, data are XORed bit by bit with a long, one-time bit stream that is generated by an initialization vector in a looping process. The looping process, as Figure 29.19 shows, generates a 64-bit sequence that is XORed with plaintext to create ciphertext.

Figure 29.19 *CSM*

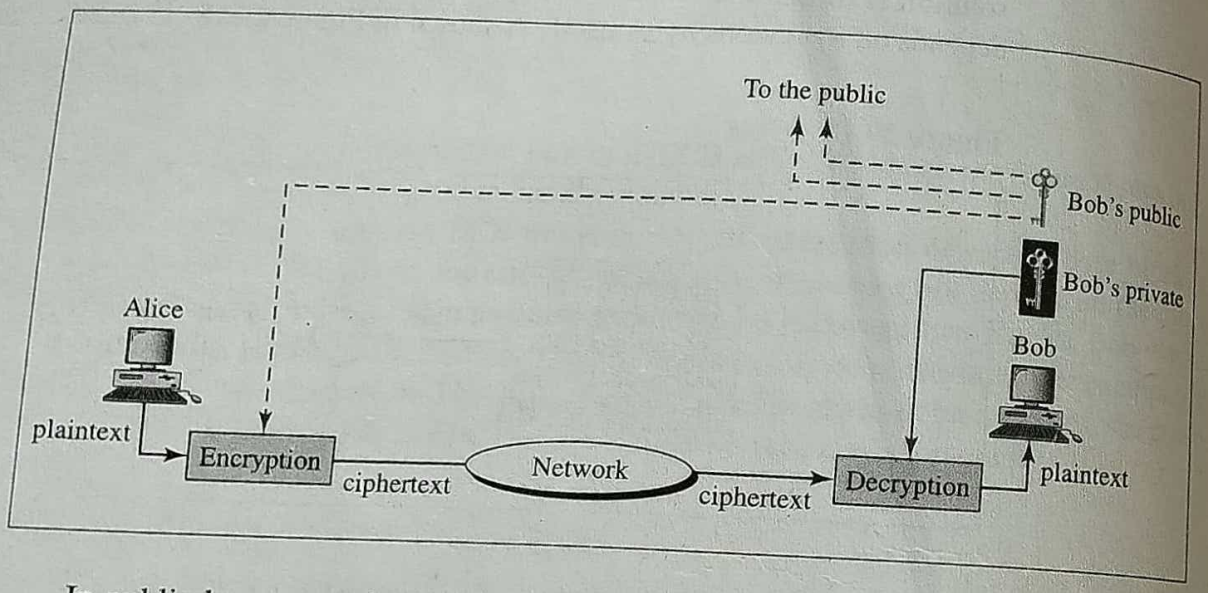


29.3 PUBLIC-KEY CRYPTOGRAPHY

In **public-key cryptography,** there are two keys: a **private key** and a **public key.** The private key is kept by the receiver. The public key is announced to the public.

Imagine Alice, as shown in Figure 29.20, wants to send a message to Bob. Alice uses the public key to encrypt the message. When the message is received by Bob, the private key is used to decrypt the message.

Figure 29.20 *Public-key cryptography*



In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption. The public key is available to the public; the private key is available only to an individual.

Public-key encryption/decryption has two advantages. First, it removes the restriction of a shared symmetric key between two entities (e.g., persons) who need to communicate with each other. A shared symmetric key is shared by the two parties and cannot be used when one of them wants to communicate with a third party. In public-key encryption/decryption, each entity creates a pair of keys; the private one is kept, and the public one is distributed. Each entity is independent, and the pair of keys created can be used to communicate with any other entity. The second advantage is that the number of keys needed is reduced tremendously. In this system, for 1 million users to communicate, only 2 million keys are needed, not 500 billion, as was the case in symmetric-key cryptography.

Public-key cryptography also has two disadvantages. The big disadvantage is the complexity of the algorithm. If we want the method to be effective, the algorithm needs large numbers. Calculating the ciphertext from plaintext using the long keys takes a lot of time. That is the main reason that public-key cryptography is not recommended for large amounts of text.

Public-key algorithms are more efficient for short messages.

The second disadvantage of the public-key method is that the association between an entity and its public key must be verified. If Alice sends her public key via an email to Bob, then Bob must be sure that the public key really belongs to Alice and nobody else.

We will see that this certification is really important when we use public-key cryptography for authentication. However, this disadvantage can be overcome using a certification authority (CA) that we discuss in Chapter 30. Public-key encryption methods are relatively new. Several methods have been used during the last few decades, but the one most common today is based on the RSA algorithm.

RSA

The most common public-key algorithm is called the **RSA method** after its inventors (Rivest, Shamir, and Adleman). The private key here is a pair of numbers (N, d); the public key is also a pair of numbers (N, e). Note that N is common to the private and public keys.

The sender uses the following algorithm to encrypt the message:

$$C = P^e \bmod N$$

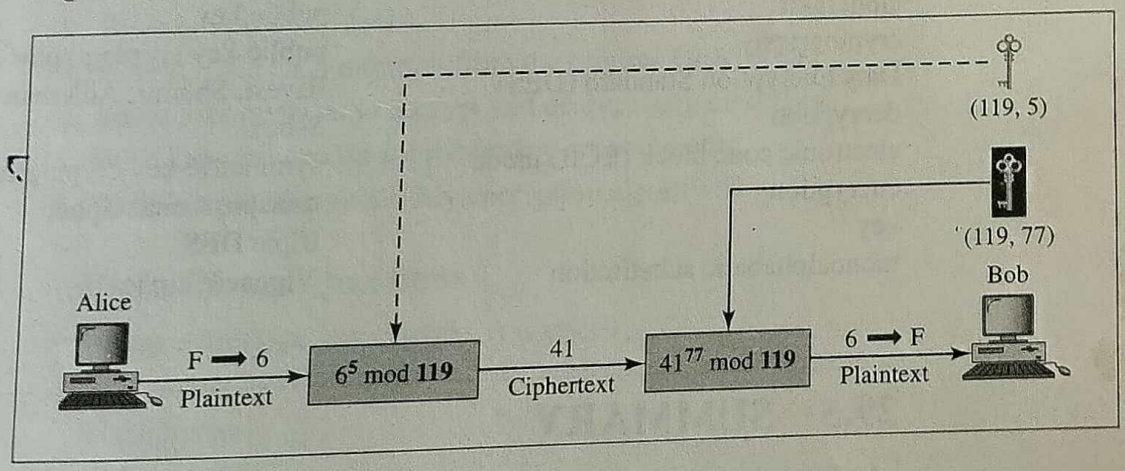
In this algorithm, P is the plaintext, which is represented as a number; C is the number that represents the ciphertext. The two numbers e and N are components of the public key. Plaintext P is raised to the power e and divided by N. The mod term indicates that the remainder is sent as the ciphertext.

The receiver uses the following algorithm to decrypt the message:

$$P = C^d \bmod N$$

In this algorithm, P and C are the same as before. The two numbers d and N are components of the private key. Figure 29.21 shows an example.

Figure 29.21 *RSA*



Imagine the private key is the pair (119, 77) and the public key is the pair (119, 5). The sender needs to send the character F. This character can be represented as number 6 (F is the sixth character in the alphabet). The encryption algorithm calculates $C = 6^5 \bmod 119 = 41$. This number is sent to the receiver as the ciphertext. The receiver uses the decryption algorithm to calculate $P = 41^{77} \bmod 119 = 6$ (the original number). The number 6 is then interpreted as F.

The reader may question the effectiveness of this algorithm. If an intruder knows the decryption algorithm and $N = 119$, the only thing missing is $d = 77$. Why couldn't

the intruder use trial and error to find d? The answer is yes, in this trivial example an intruder could easily guess the value of d. But a major concept of the RSA algorithm is to use very large numbers for d and e. In practice, the numbers are so large (on the scale of tens of digits) that the trial-and-error approach of breaking the code takes a long time (years, if not months) even with the fastest computers available today.

Choosing Public and Private Keys

One question that comes to mind is, How do we choose the three numbers N, d, and e for encryption and decryption to work? The inventors of the RSA used number theory to prove that using the following procedure will guarantee that the algorithms will work. Although the proof is beyond the scope of this book, we outline the procedure:

1. Choose two large prime numbers p and q.
2. Compute $N = p \times q$.
3. Choose e (less than N) such that e and $(p - 1)(q - 1)$ are relatively prime (having no common factor other than 1).
4. Choose d such that $(e \times d)$ mod $[(p - 1)(q - 1)]$ is equal to 1.

29.4 KEY TERMS

block cipher
cipher
cipher block chaining (CBC) mode
cipher feedback mode (CFM)
cipher stream mode (CSM)
ciphertext
cryptography
Data Encryption Standard (DES)
decryption
electronic code block (ECB) mode
encryption
key
monoalphabetic substitution
P-box
plaintext
polyalphabetic substitution
private key
product block
public key
public-key cryptography
Rivest, Shamir, Adleman (RSA) method
S-box
symmetric-key cryptography
transpositional cipher
triple DES
Vigenere cipher

29.5 SUMMARY

- Cryptography is the science and art of transforming messages to make them secure and immune to attack.
- Encryption renders a message (plaintext) unintelligible to unauthorized personnel.
- Decryption transforms an intentionally unintelligible message (ciphertext) into meaningful information.
- Cryptography algorithms are classified as either symmetric-key methods or public-key methods.

- ❑ In symmetric-key cryptography the same secret key is used by the sender and the receiver.
- ❑ Substitution ciphers are either monoalphabetic or polyalphabetic.
- ❑ The P-box, S-box, and product block are methods used by block ciphers.
- ❑ DES is a symmetric-key method adopted by the U.S. government, but it has been replaced by Triple DES or other methods.
- ❑ Operation modes to handle long messages include ECB mode, CBC mode, CFM, and CSM.
- ❑ In public-key cryptography, the public key is used by the sender to encrypt the message; the private key is used by the receiver to decrypt the message.
- ❑ One of the commonly used public-key cryptography methods is the RSA algorithm.

29.6 PRACTICE SET

Review Questions

1. What is the relationship between plaintext and ciphertext?
2. What are the two categories of cryptography methods? What is the main difference between the categories?
3. What is the concept behind substitutional cryptography?
4. Why is polyalphabetic substitution superior to monoalphabetic substitution?
5. What is the concept behind transpositional cryptography?
6. What is a block cipher?
7. What is the function of a P-box?
8. What is a product block?
9. How is triple DES different from the original DES?
10. Name four methods to encrypt and decrypt long messages.
11. What keys are needed for public-key cryptography?
12. What is a popular public-key encryption algorithm?

Multiple-Choice Questions

13. Before a message is encrypted, it is called _____.
    a. Plaintext
    b. Ciphertext
    c. Cryptotext
    d. Cryptonite
14. After a message is decrypted, it is called _____.
    a. Plaintext
    b. Ciphertext
    c. Cryptotext
    d. Cryptonite

...divided in...
...ypted independent...

...ES (Advanced Encry... Standard) :-

AES is a widely used blo... cipher that encrypt...
blocks of 128 bits using a key of 128, 192 or 256

15. A cipher is _____.
    a. An encryption algorithm
    b. A decryption algorithm
    c. A private key
    d. (a) or (b)
16. In the symmetric-key method of cryptography, which key is publicly known?
    a. Encryption key only
    b. Decryption key only
    c. Both
    d. None of the above
17. If 20 people need to communicate using symmetric-key cryptography, _____ symmetric keys are needed.
    a. 19
    b. 20
    c. 190
    d. 200
18. The _____ is an example of polyalphabetic substitution.
    a. P-box
    b. S-box
    c. Product block
    d. Vigenere cipher
19. The _____ is a block cipher.
    a. P-box
    b. S-box
    c. Product block
    d. All the above
20. We use a cryptography method in which the character Z always substitutes for the character G. This is probably _____.
    a. Monoalphabetic substitution
    b. Polyalphabetic substitution
    c. Transpositional
    d. None of the above

We use an cryptography method in which the plaintext AAAAAA becomes the ciphertext BCDEFG. This is probably _____.
a. Monoalphabetic substitution
b. Polyalphabetic substitution
c. Transposition
d. None of the above

One way to encrypt and decrypt long messages is through the use of the _____.
a. ECB mode
b. CBC mode

c. CFM
d. All the above

23. An initialization vector is needed in the _____.
a. ECB mode
b. CBC mode
c. CVF
d. CSM

24. In the _____ the eneryption of each 8-byte block is independent of the others.
a. ECB mode
b. CBC mode
c. CVF
d. CSM

25. In the public-key method of cryptography, which key is publicly known?
a. Eneryption key only
b. Decryption key only
c. Both
d. None of the above

26. In the public-key method of cryptography, only the receiver has possession of the _____.
a. Private key
b. Public key
c. Both keys
d. None of the above

27. The RSA algorithm uses a _____ cryptography method.
a. Public-key
b. Private-key
c. Symmetric-key
d. Denominational

Exercises

28. Encrypt the following message, using monoalphabetic substitution with key = 4.
THIS IS A GOOD EXAMPLE

29. Decrypt the following message, using monoalphabetic substitution with key = 4.
IRGVCTXMSR MW JYR

30. Decrypt the following message, using monoalphabetic substitution without knowing the key.
KTIXEVZOUT OY ROQK KTIRUYOTM G YKIXKZ OT GT KTBKRUVK

31. Encrypt the following message, using polyalphabetic substitution. Use the position of each character as the key.
One plus one is two, one plus two is three, one plus three is four.

32. Use the following encrypting algorithms to encrypt the message "GOOD DAY."
    a. Replace each character with its ASCII code.
    b. Add a 0 bit at the left to make each character 8 bits long.
    c. Swap the first 4 bits with the last 4 bits.
    d. Replace every 4 bits with its hexadecimal equivalent.

    What is the key in this method?

33. Use the following encrypting algorithm to encrypt the message "ABCADEFGH" (assume that the message is always made of uppercase letters).
    a. Treat each character as a decimal number, using ASCII code (between
    a. 65 and 90).
    b. Subtract 65 from each coded character.
    c. Change each number into a 5-bit pattern.

34. Using the RSA algorithm, encrypt and decrypt the message "BE" with key pairs (3, 15) and (5, 15).

35. Given the two prime numbers $p = 19$ and $q = 23$, try to find N, e, and d.

36. To understand the security of the RSA algorithm, find d if you know that $e = 17$ and $N = 187$.

37. In the RSA algorithm, we use $C = P^e \bmod N$ to encrypt a number. If e and N are large numbers (each hundreds of digits), the calculation is impossible and creates an overflow error even in a supercomputer. One solution (not the best one) using number theory involves several steps, where each step uses the result of the previous step:
    a. $C = 1$.
    b. Repeat e times:

$$C = (C \times P) \bmod N$$

In this way, a computer program can be written that calculates C using a loop. For example $6^5 \bmod 119$, which is 41, can be calculated as follows:

$$(1 \times 6) \bmod 119 = 6$$
$$(6 \times 6) \bmod 119 = 36$$
$$(36 \times 6) \bmod 119 = 97$$
$$(97 \times 6) \bmod 119 = 106$$
$$(106 \times 6) \bmod 119 = 41$$

Use this method to calculate $227^{16} \bmod 100$.

C → 1 2 3 4 5 6
Y A U

Y A
X Y - - -
D Z . .

Y × D ADZ

CHAPTER 30

Message Security, User Authentication, and Key Management

After studying cryptography in Chapter 29, we discuss some of its applications: message security, user authentication, and key management.

Message security involves confidentiality, integrity, authentication, and finally nonrepudiation.

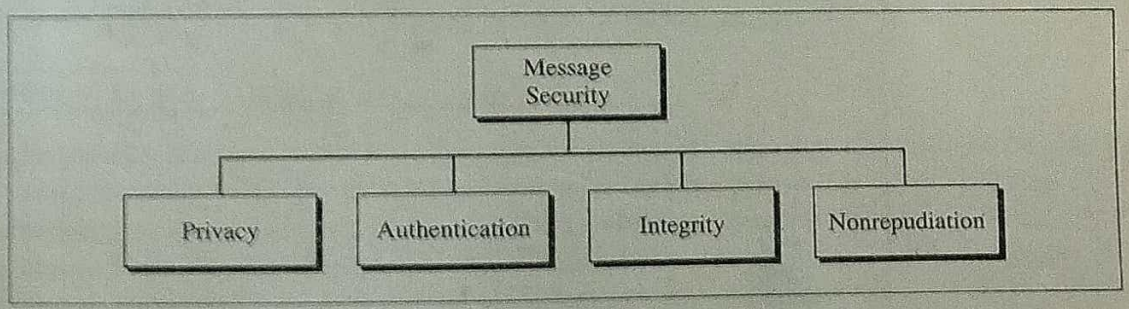
User authentication means verifying the identity of the person or process that wants to communicate with a system. User authentication is also needed for key management.

Finally, we need key management: the distribution of symmetric keys and the certification of the public keys. Section 30.4 explains the methods used in key management.

30.1 MESSAGE SECURITY

Let us first discuss the security measures applied to each single message. We can say that security provides four services: privacy (confidentiality), message authentication, message integrity, and nonrepudiation (see Fig. 30.1).

Figure 30.1 *Message security*



Privacy

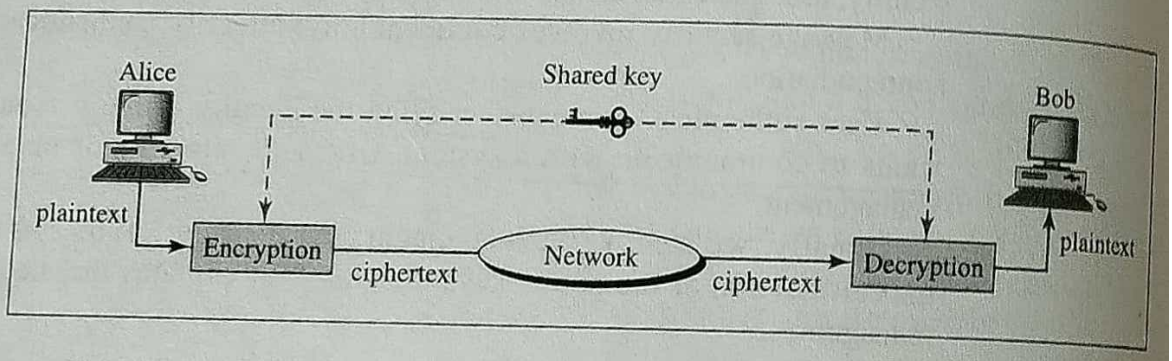
Privacy means that the sender and the receiver expect confidentiality. The transmitte message must make sense to only the intended receiver. To all others, the messag must be unintelligible.

The concept of how to achieve privacy has not changed for thousands of years The message must be encrypted. That is, the message must be rendered unintelligible to unauthorized parties. A good privacy technique guarantees to some extent that a potential intruder (eavesdropper) cannot understand the contents of the message.

Privacy with Symmetric-Key Cryptography

Privacy can be achieved using symmetric-key encryption and decryption, as shown in Figure 30.2. As we discussed in Chapter 29, in symmetric-key cryptography the key is shared between Alice and Bob.

Figure 30.2 *Privacy using symmetric-key encryption*

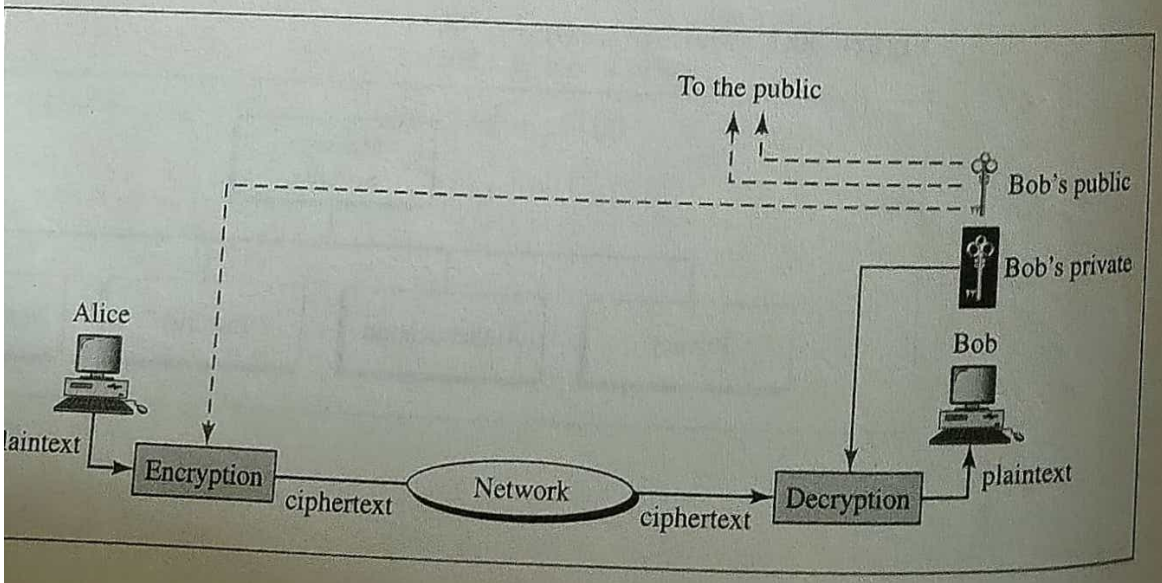


Using symmetric-key cryptography is very common for achieving privacy. Later in this chapter, we will see how to manage the distribution of symmetric keys.

Privacy with Public-Key Cryptography

We can also achieve privacy using public-key encryption. There are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public. This is shown in Figure 30.3.

Figure 30.3 *Privacy using public-key encryption*



with public key encryption is its owner must be verified (certi-
o solve this problem shortly.

cation (Identity of Sender)

ion means that the receiver needs to be sure of the sender's iden-
ster has not sent the message. We will see how digital signature
authentication.

Data Unaltered)

hat the data must arrive at the receiver exactly as they were sent.
changes during the transmission, either accidental or malicious.
re monetary exchanges occur over the Internet, integrity is crucial.
would be disastrous if a request for transferring $100 changed to a
,000 or $100,000. The integrity of the message must be preserved in
unication. We will see how digital signature can provide message

ation have proof of identity of the sender

tion means that a receiver must be able to prove that a received message
specific sender. The sender must not be able to deny sending a message that
fact, did send. The burden of proof falls on the receiver. For example, when
sends a message to transfer money from one account to another, the bank
proof that the customer actually requested this transaction. We will see how
nature can provide nonrepudiation.

DIGITAL SIGNATURE

d that security provides four services in relation to a single message: privacy,
tication, integrity, and nonrepudiation. We have already discussed privacy. The
three can be achieved by using what is called **digital signature.**
The idea is similar to the signing of a document. When we send a document elec-
cally, we can also sign it. We have two choices: We can sign the entire document,
e can sign a digest (condensed version) of the document.

gning the Whole Document

ublic-key encryption can be used to sign a document. However, the roles of the public
nd private keys are different here. The sender uses her private key to encrypt (sign) the
nessage just as a person uses her signature (which is private in the sense that it is diffi-
cult to forge) to sign a paper document. The receiver, on the other hand, uses the public
key of the sender to decrypt the message just as a person verifies from memory another
person's signature.

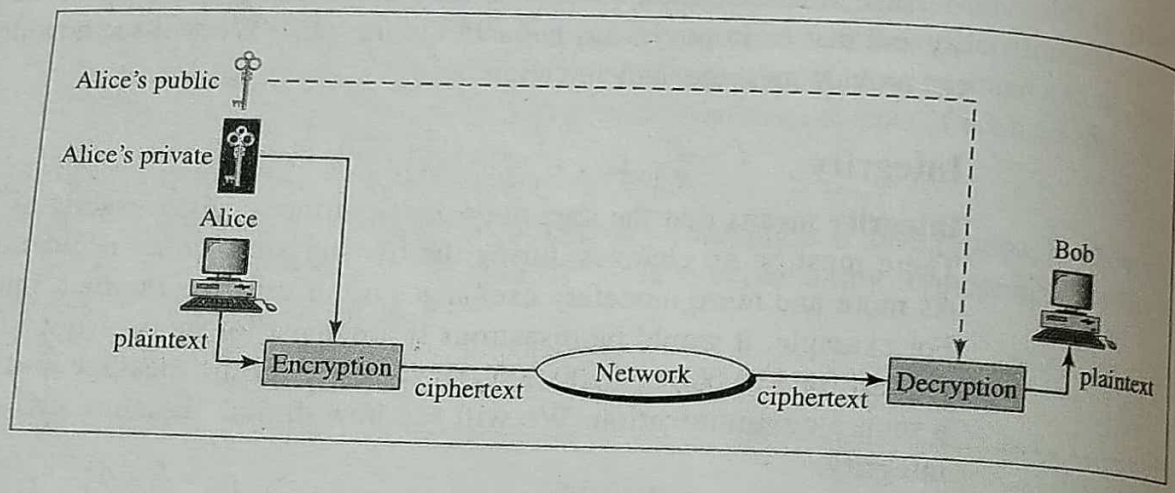
In the digital signature, the private key is used for encryption and the public key for
decryption. This is possible because the encryption and decryption algorithms used today,

avided
ypted independen

(Advanced Encry n standard) :-
AES is a widely used blo ipher that encryp
blocks of 128 bits using a key of 128, 192 or 25

such as RSA, are mathematical formulas and their structures are similar. Figure 30.4 shows how this is done.

Figure 30.4 *Signing the whole document*



Digital signatures can provide integrity, authentication, and nonrepudiation.

Integrity The integrity of a message is preserved because if Eve intercepted the message and partially or totally changed it, the decrypted message would be unreadable.

Authentication We can use the following reasoning to show how a message can be authenticated. If Eve sends a message while pretending that it is coming from Alice, she must use her own private key for encryption. The message is then decrypted with the public key of Alice and will therefore be nonreadable. Encryption with Eve's private key and decryption with Alice's public key result in garbage.

Nonrepudiation Digital signature also provides for nonrepudiation. Bob saves the message received from Alice. If Alice later denies sending the message, Bob can show that encrypting and decrypting the saved message with Alice's private and public key can create a duplicate of the saved message. Since only Alice knows her private key, she cannot deny sending the message.

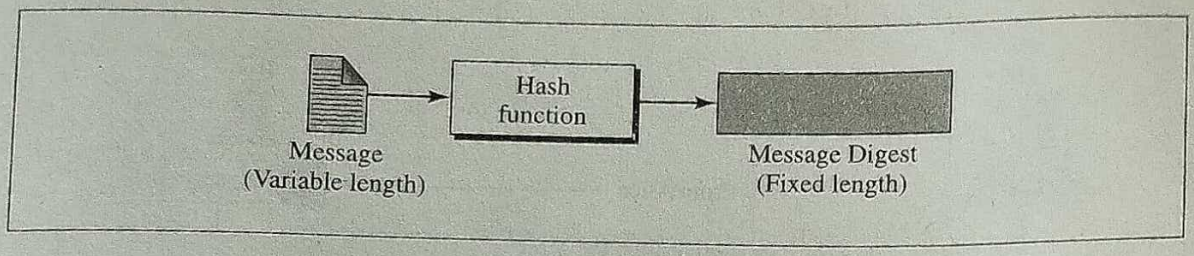
Digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

Signing the Digest

We said before that public-key encryption is efficient if the message is short. Using a public key to sign the entire message is very inefficient if the message is very long. The solution is to let the sender sign a digest of the document instead of the whole document. The sender creates a miniature version or **digest** of the document and signs it; the receiver then checks the signature on the miniature.

To create a digest of the message, we use a **hash function.** The hash function creates a fixed-size digest from a variable-length message, as shown in Figure 30.5.

Figure 30.5 *Signing the digest*

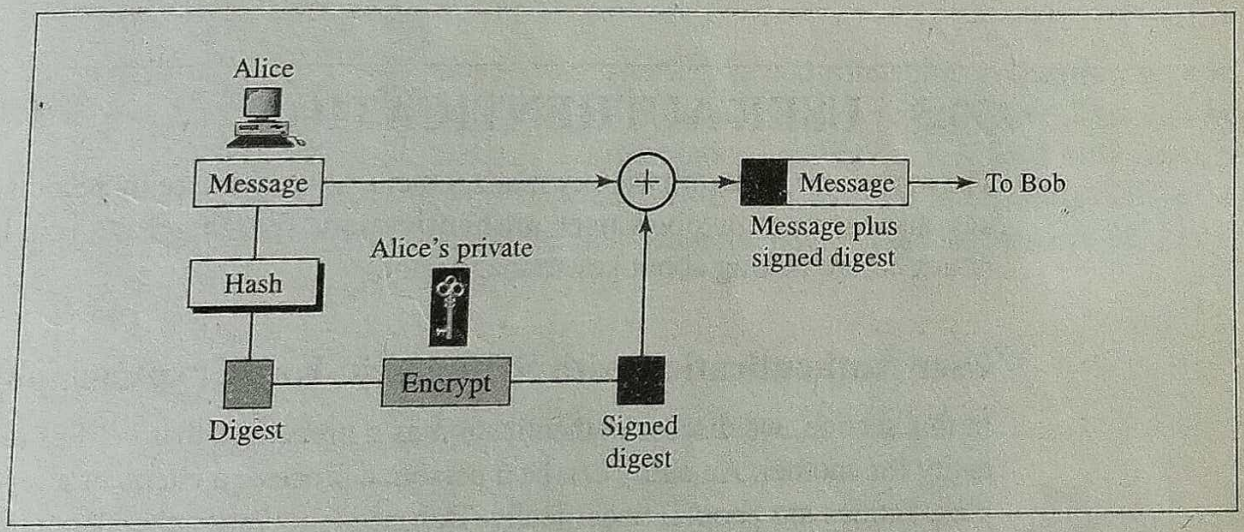


The two most common hash functions are called MD5 (Message Digest 5) and SHA-1 (Secure Hash Algorithm 1). The first one produces a 120-bit digest. The second produces a 160-bit digest.

Note that a hash function must have two properties to guarantee its success. First, hashing is one-way; the digest can only be created from the message, not vice versa. Second, hashing is a one-to-one function; there is little probability that two messages will create the same digest. We will see the reason for this condition shortly.

After the digest has been created, it is encrypted (signed) using the sender's private key. The encrypted digest is attached to the original message and sent to the receiver. Figure 30.6 shows the sender site.

Figure 30.6 *Sender site*

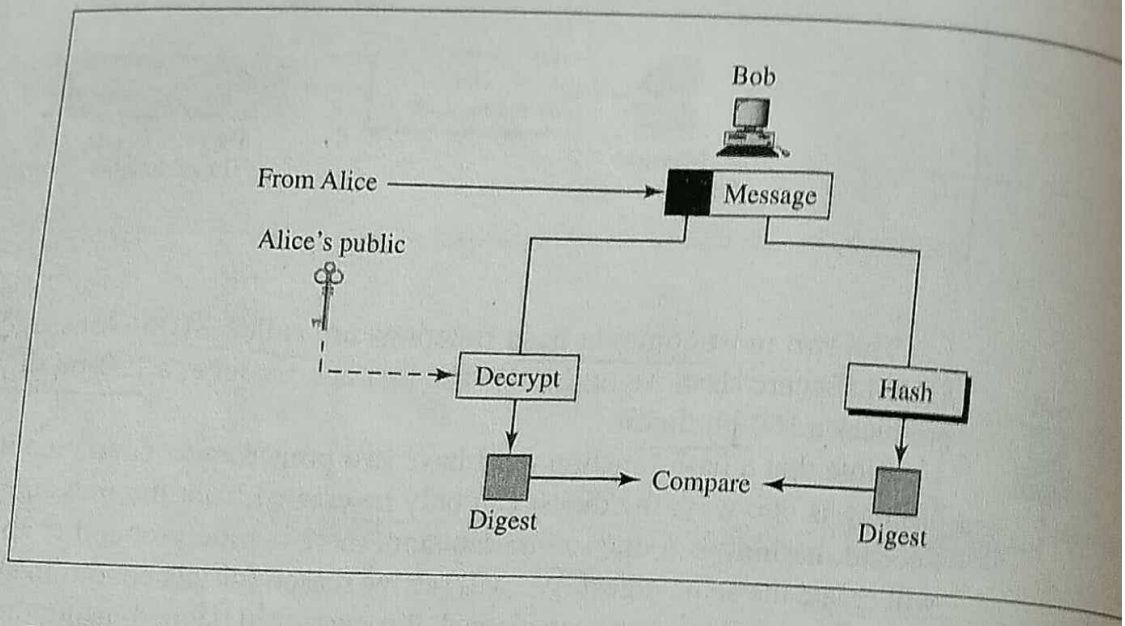


The receiver receives the original message and the encrypted digest. He separates the two. He applies the same hash function to the message to create a second digest. He also decrypts the received digest, using the public key of the sender. If the two digests are the same, all three security measures are preserved. Figure 30.7 shows the receiver site.

According to Section 30.1, we know that the digest is secure in terms of integrity, authentication, and nonrepudiation, but what about the message itself? The following reasoning shows that the message itself is also secured:

1. The digest has not been changed (integrity), and the digest is a representation of the message. So the message has not been changed (remember, it is improbable that two messages create the same digest). Integrity has been provided.

Figure 30.7 *Receiver site*



2. The digest comes from the true sender, so the message also comes from the true sender. If an intruder had initiated the message, the message would not have created the same digest (it is improbable that two messages create the same digest).
3. The sender cannot deny the message since she cannot deny the digest; the only message that can create that digest, with a very high probability, is the received message.

30.3 USER AUTHENTICATION

The main issue in security is key management, as we will see in Section 30.4. However, key management involves **user authentication.** We, therefore, briefly discuss these issues before talking about key management.

User Authentication with Symmetric-Key Cryptography

In this section, we discuss authentication as a procedure that verifies the identity of one entity for another. An *entity* can be a person, a process, a client, or a server; in our examples, entities are people. Specifically, Bob needs to verify the identity of Alice and vice versa. Note that entity authentication, as discussed here, is different from the message authentication that we discussed in the previous section. In message authentication, the identity of the sender is verified for each single message. In user authentication, the user identity is verified once for the entire duration of system access.

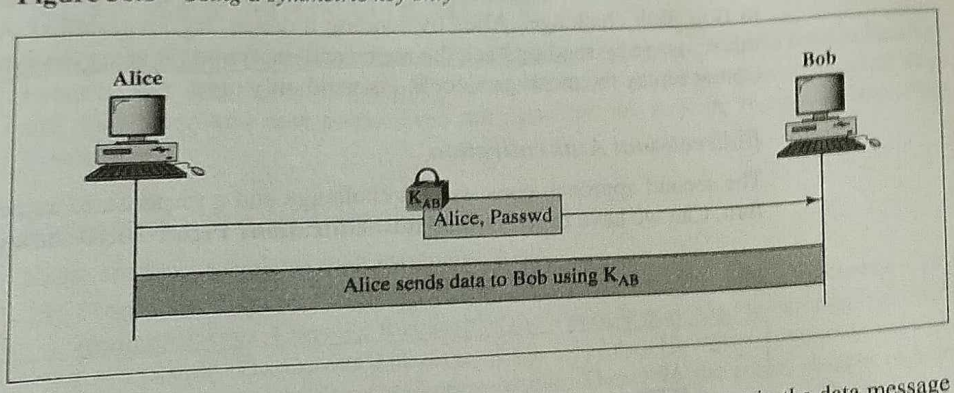
First Approach

padlock - Sept, 2019

In the first approach, Alice sends her identity and password in an encrypted message, using the symmetric key K_{AB}. Figure 30.8 shows the procedure. We have added the padlock with the corresponding key (shared key between Alice and Bob) to show that the message is encrypted with the key.

Is this a safe approach? Yes, to some extent. Eve, the intruder, cannot decipher the password or the data because she does not know K_{AB}. However, Eve can cause damage

Figure 30.8 *Using a symmetric key only*

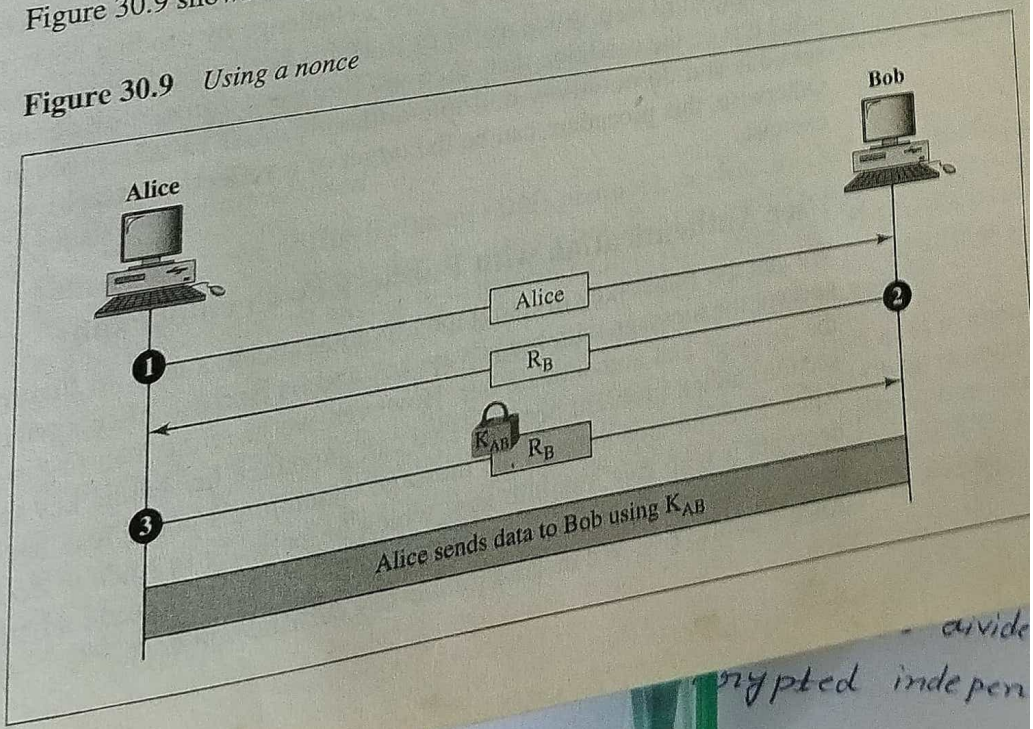


without accessing the contents of the message. If Eve has an interest in the data message sent from Alice to Bob, she can intercept both the authentication message and the data message, store them, and resend them later to Bob. Bob has no way to know that this is a replay of a previous message. There is nothing in this procedure to guarantee the freshness of the message. As an example, suppose Alice's message instructs Bob (as a bank manager) to pay Eve for some job she has done. Eve can resend the message, thereby illegally getting paid twice for the same job. This is called a **replay attack.**

Second Approach

To prevent a replay attack (or playback attack), we add something to the procedure to help Bob distinguish a fresh authentication request from a repeated one. This can be done by using a **nonce.** A nonce is a large random number that is used only once, a one-time number. In this second approach, Bob uses a nonce to challenge Alice, to make sure that Alice is authentic and that someone (Eve) is not impersonating Alice. Figure 30.9 shows the procedure.

Figure 30.9 *Using a nonce*



- divide
nypted indepen

(Advanced Encr n standard) :-

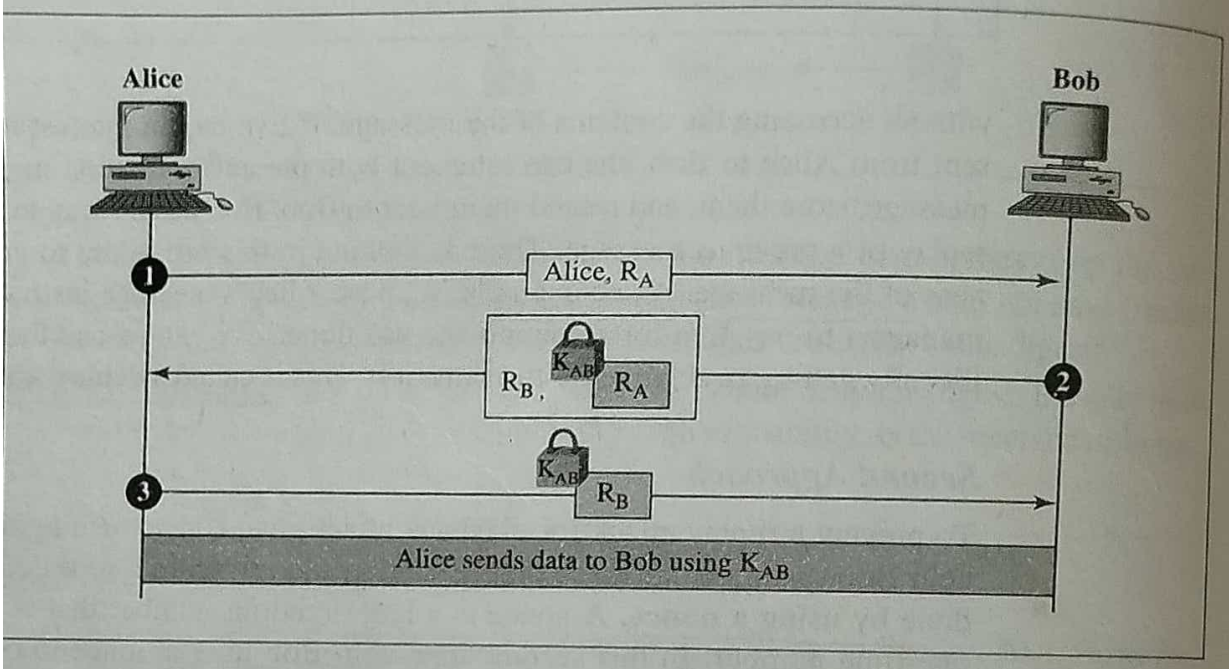
AES is a widely used bl eipher that enc
blocks of 128 bits using a key of 128, 192 or

Authentication happens in three steps. First, Alice sends her identity, in plaintext, o Bob. Bob challenges Alice by sending a nonce, R_B, in plaintext. Alice responds to his message by sending back the nonce and encrypting it using the symmetric key. Eve annot replay the message since R_B is valid only once.

Bidirectional Authentication

The second approach consists of a challenge and a response to authenticate Alice for Bob. Can we have **bidirectional authentication?** Figure 30.10 shows one method.

Figure 30.10 *Bidirectional authentication*



In the first step, Alice sends her identification and her nonce to challenge Bob. In he second step, Bob responds to Alice's challenge by sending his nonce to challenge ier. In the third step, Alice responds to Bob's challenge. Is this authentication totally afe? It is on the condition that Alice and Bob use a different set of nonces for different essions and do not allow multiple authentications to take place at the same time. Otherwise, this procedure can be the target of a **reflection attack;** we leave this as an xercise.

User Authentication with Public-Key Cryptography

We can use public-key cryptography to authenticate a user. In Figure 30.9, Alice can ncrypt the message with her private key and let Bob use Alice's public key to decrypt he message and authenticate her. However, we have the man-in-the-middle (see next ection) attack problem because Eve can announce her public key to Bob in place of Alice. Eve can then encrypt the message containing a nonce with her private key. Bob lecrypts it with Eve's public key, which he believes is Alice's. Bob is fooled. Alice eeds a better means to advertise her public key; Bob needs a better way to verify lice's public key. We discuss public-key certification next.

30.4 KEY MANAGEMENT

We discussed how symmetric-key and public-key cryptography can be used in message security and user authentication. However, we never explained how symmetric keys are distributed and how public keys are certified. We explore these two important issues here.

Symmetric Key Distribution

There are three problems with symmetric keys.

1. First, if n people want to communicate with one another, there is a need for $n(n-1)/2$ symmetric keys. Consider that each of the n people may need to communicate with $n-1$ people. This means that we need $n(n-1)$ keys. However, symmetric keys are shared between two communicating people. Therefore, the actual number of keys needed is $n(n-1)/2$. This is usually referred to as the **n^2 problem.** If n is a small number, this is acceptable. For example, if 5 people need to communicate, only 10 keys are needed. The problem is aggravated if n is a large number. For example, if n is 1 million, almost half a trillion keys are needed.
2. Second, in a group of n people, each person must have and remember $n-1$ keys, one for every other person in the group. This means that if 1 million people want to communicate with one another, each must remember (or store) almost 1 million keys in his or her computer.
3. Third, how can two parties securely acquire the shared key? It cannot be done over the phone or the Internet; these are not secure.

Session Keys

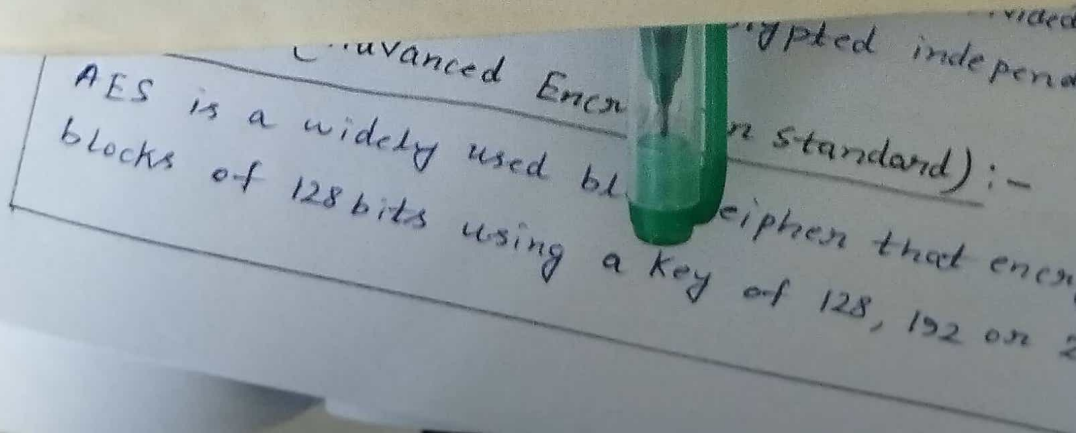
Considering the above problems, a symmetric key between two parties is useful if it is dynamic: created for each session and destroyed when the session is over. It does not have to be remembered by the two parties.

A symmetric key between two parties is useful if it is used only once; it must be created for one session and destroyed when the session is over.

Diffie-Hellman Method

One protocol, the **Diffie-Hellman (DH) protocol,** devised by Diffie and Hellman, provides a one-time session key for two parties. The two parties use the session key to exchange data without having to remember or store it for future use. The parties do not have to meet to agree on the key, it can be done through the Internet. Let us see how the protocol works when Alice and Bob need a symmetric key to communicate.

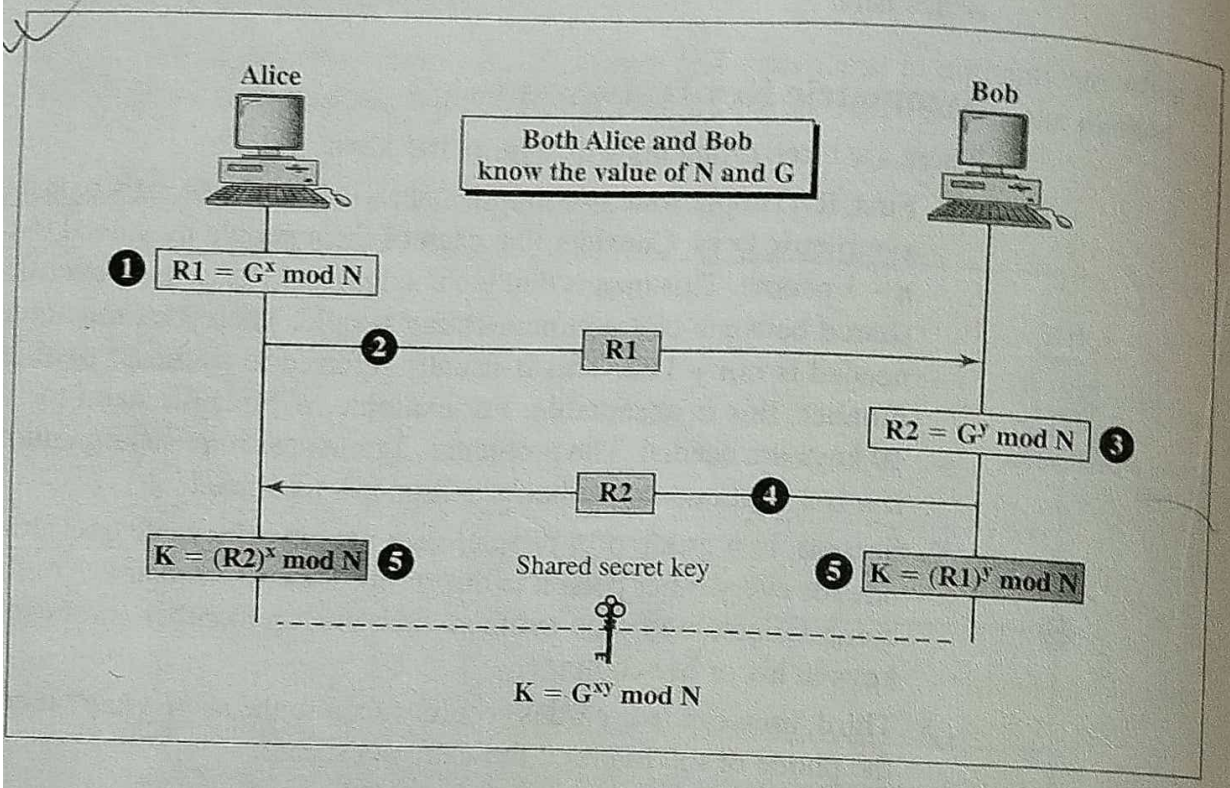
Prerequisite Before establishing a symmetric key, the two parties need to choose two numbers N and G. The first number, N, is a large prime number with restriction that (N – 1)/2 must also be a prime number; the second number G is also a prime number, but it has more restrictions. These two numbers need not be confidential. They can be sent through the Internet; they can be public. Any two numbers, selected properly



can serve the entire world. There is no secrecy about these two numbers; both Alice and Bob know these magic numbers.

Procedure Figure 30.11 shows the procedure.

Figure 30.11 *Diffie-Hellman method*



The steps are as follows:

Step 1 Alice chooses a large random number x and calculates $R1 = G^x \bmod N$.

Step 2 Alice sends R1 to Bob. Note that Alice does not send the value of x; she only sends R1.

Step 3 Bob chooses another large number y and calculates $R2 = G^y \bmod N$.

Step 4 Bob sends R2 to Alice. Again, note that Bob does not send the value of y; he only sends R2.

Step 5 Alice calculates $K = (R2)^x \bmod N$. Bob also calculates $K = (R1)^y \bmod N$. And K is the symmetric key for the session.

he reader may wonder why the value of K is the same since the calculations are fferent. The answer is an equality proved in number theory.

$$(G^x \bmod N)^y \bmod N = (G^y \bmod N)^x \bmod N = G^{xy} \bmod N$$

Bob has calculated $K = (R1)^y \bmod N = (G^x \bmod N)^y \bmod N = G^{xy} \bmod N$. Alice calculated $K = (R2)^x \bmod N = (G^y \bmod N)^x \bmod N = G^{xy} \bmod N$. Both have reached same value without Bob knowing the value of x or Alice knowing the value of y.

he symmetric (shared) key in the Diffie-Hellman protocol is $K = G^{xy} \bmod N$.

Example 1

Let us give an example to make the procedure clear. Our example uses small numbers, but note that in a real situation, the numbers are very large. Assume G = 7 and N = 23. The steps are as follows:

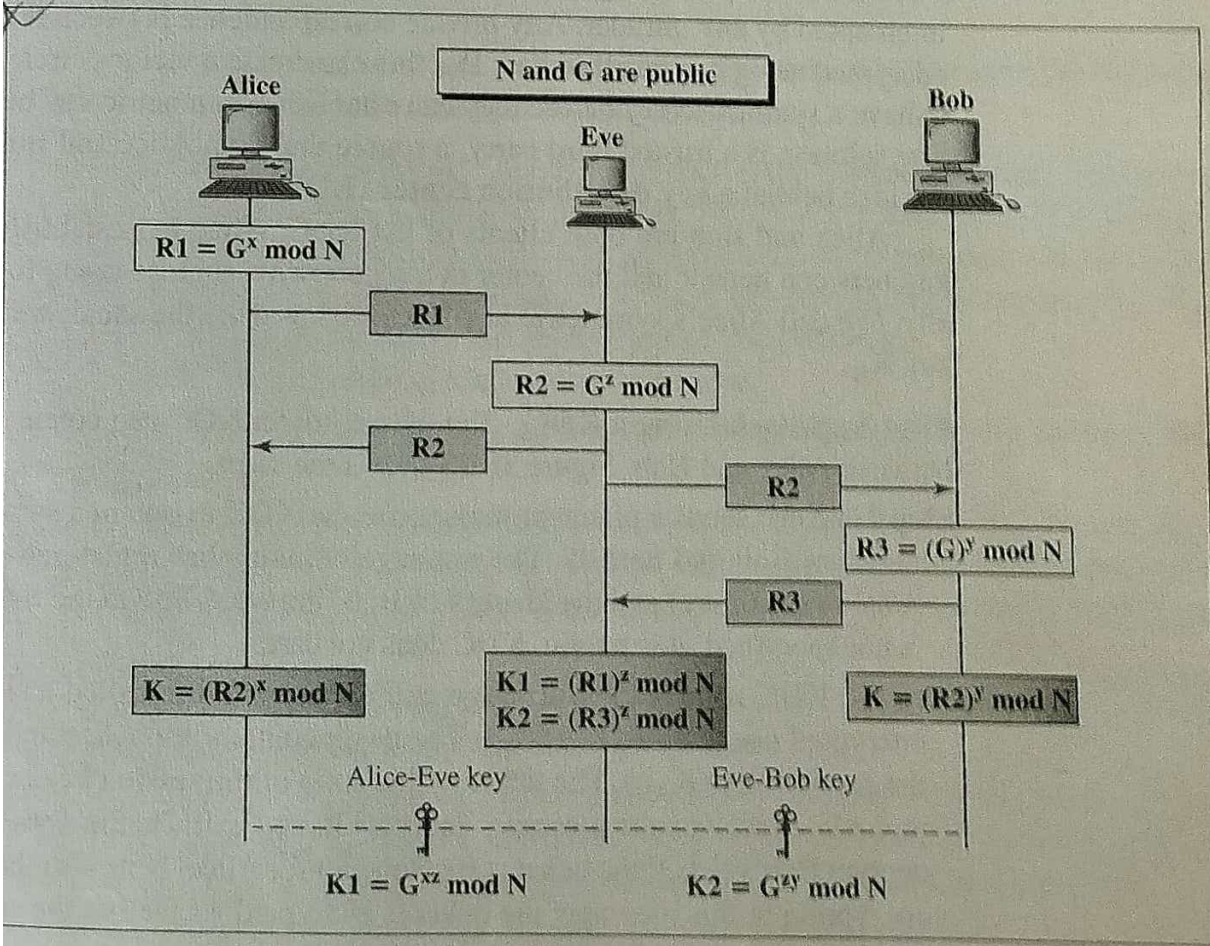
1. Alice chooses x = 3 and calculates $R1 = 7^3 \bmod 23 = 21$.
2. Alice sends the number 21 to Bob.
3. Bob chooses y = 6 and calculates $R2 = 7^6 \bmod 23 = 4$.
4. Bob sends the number 4 to Alice.
5. Alice calculates the symmetric key $K = 4^3 \bmod 23 = 18$.
6. Bob calculates the symmetric key $K = 21^6 \bmod 23 = 18$.

The value of K is the same for both Alice and Bob; $G^{xy} \bmod N = 7^{18} \bmod 23 = 18$.

Man-in-the-Middle Attack The Diffie-Hellman protocol is a very sophisticated symmetric-key creation algorithm. If x and y are very large numbers, it is extremely difficult for Eve to find the key knowing only N and G. An intruder needs to determine x and y if R1 and R2 are intercepted. But finding x from R1 and y from R2 are two difficult tasks. Even a sophisticated computer would need perhaps a long time to find the key by trying different numbers. In addition, Alice and Bob change the key the next time they need to communicate.

However, the protocol does have a weakness. Eve does not have to find the values of x and y to attack the protocol. She can fool Alice and Bob by creating two keys: one between herself and Alice and another between herself and Bob. Figure 30.12 shows the situation.

Figure 30.12 *Man-in-the-middle attack*



The following can happen:

1. Alice chooses x, calculates $R1 = G^x \bmod N$, and sends R1 to Bob.
2. Eve, the intruder, intercepts R1. She chooses z, calculates $R2 = G^z \bmod N$, and sends R2 to both Alice and Bob.
3. Bob chooses y, calculates $R3 = G^y \bmod N$, and sends R3 to Alice. R3 is intercepted by Eve and never reaches Alice.
4. Alice and Eve calculate $K1 = G^{xz} \bmod N$, which becomes a shared key between Alice and Eve. Alice, however, thinks that it is a key shared between Bob and herself.
5. Eve and Bob calculate $K2 = G^{zy} \bmod N$, which becomes a shared key between Eve and Bob. Bob, however, thinks that it is a key shared between Alice and himself.

In other words, two keys, instead of one, are created: one between Alice and Eve and one between Eve and Bob. When Alice sends data to Bob encrypted with K1 (shared by Alice and Eve), the data can be deciphered and read by Eve. Eve can send the message to Bob encrypted by K2 (shared key between Eve and Bob); or she can even change the message or send a totally new message. Bob is fooled into believing that the message has come from Alice. The same scenario can happen to Alice in the other direction.

This situation is called a **man-in-the-middle attack** because Eve comes in between and intercepts R1, sent by Alice to Bob, and R3, sent by Bob to Alice. It is also known as a bucket brigade attack because it resembles a short line of volunteers passing a bucket of water from person to person.

Key Distribution Center (KDC)

The flaw in the previous protocol is the sending of R1 and R2 as plaintext which can be intercepted by any intruder. Any private correspondence between two parties should be encrypted using a symmetric key. But this can create a vicious circle. Two parties need to have a symmetric key before they can establish a symmetric key between themselves. The solution is a trusted third party, a source that both Alice and Bob can trust. This is the idea behind a **key distribution center (KDC).**

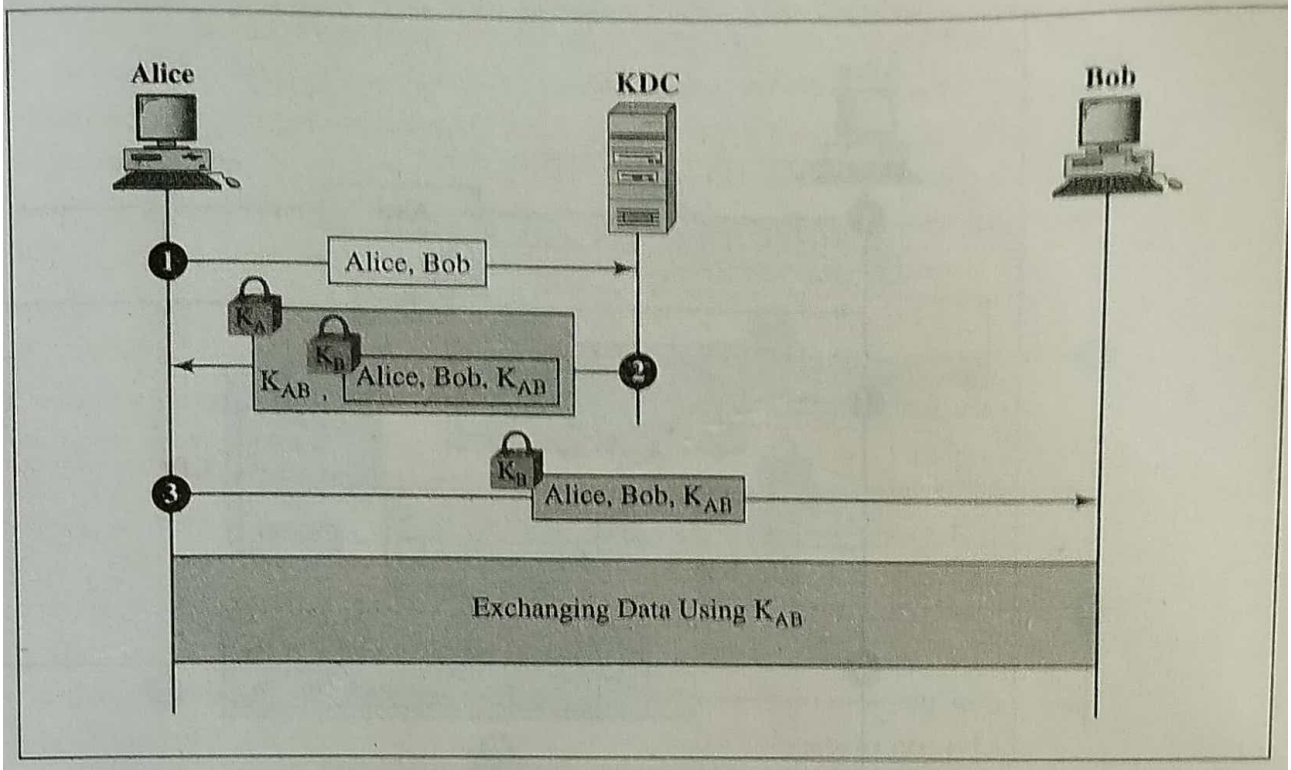
Alice and Bob are both clients of the KDC. Alice has established one symmetric key between herself and the center in a secure way, such as going to the center personally. We call Alice's symmetric key K_A. Bob has done the same; we call his symmetric key K_B.

First Approach Using a KDC Let us see how a KDC can create a session key K_{AB} between Alice and Bob. Figure 30.13 shows the steps.

Step 1 Alice sends a plaintext message to the KDC to obtain a symmetric session key between Bob and herself. The message contains her registered identity (the word *Alice* in the figure) and the identity of Bob (the word *Bob* in the figure). This message is not encrypted; it is public. KDC does not care.

Step 2 KDC receives the message and creates what is called a **ticket.** The ticket is encrypted using Bob's key (K_B). The ticket contains the Alice and Bob identities and the session key (K_{AB}). The ticket with a copy of the session key is sent to Alice. Note that Alice receives the message, decrypts it, and extracts the session key. She cannot decrypt Bob's ticket; the ticket is for Bob, not for Alice. Note also that we have a double encryption in this message; the ticket is encrypted, as well as the entire message.

Figure 30.13 *First approach using KDC*



Step 3 Alice sends the ticket to Bob. Bob opens the ticket and knows that Alice needs to send messages to him using K_{AB} as the session key.

Sending data. After the third step, Alice and Bob can exchange data using K_{AB} as a one-time session key.

Eve can use the replay attack we discussed previously. She can save the message in step 3 as well as the data messages and replay all.

Needham-Schroeder Protocol Another approach is the elegant **Needham-Schroeder protocol,** a foundation for many other protocols. This protocol uses multiple challenge-response interactions between parties to achieve a flawless protocol. In the latest version of this protocol, Needham and Schroeder use four different nonces: R_A, R_B, R_1, and R_2. Figure 30.14 shows the seven steps of this protocol.

The following are brief descriptions of each step:

Step 1 Alice sends her identity to Bob, thereby declaring that she needs to talk to him.

Step 2 Bob uses nonce R_B and encrypts it with his symmetric key K_B. Nonce R_B is intended for the KDC, but it is sent to Alice. Alice sends R_B to the KDC to prove that the person who has talked to Bob is the same person (not an imposter) who will talk to the KDC.

Step 3 Alice sends a message to the KDC that includes her nonce, R_A, her identity, Bob's identity, and the encrypted nonce from Bob.

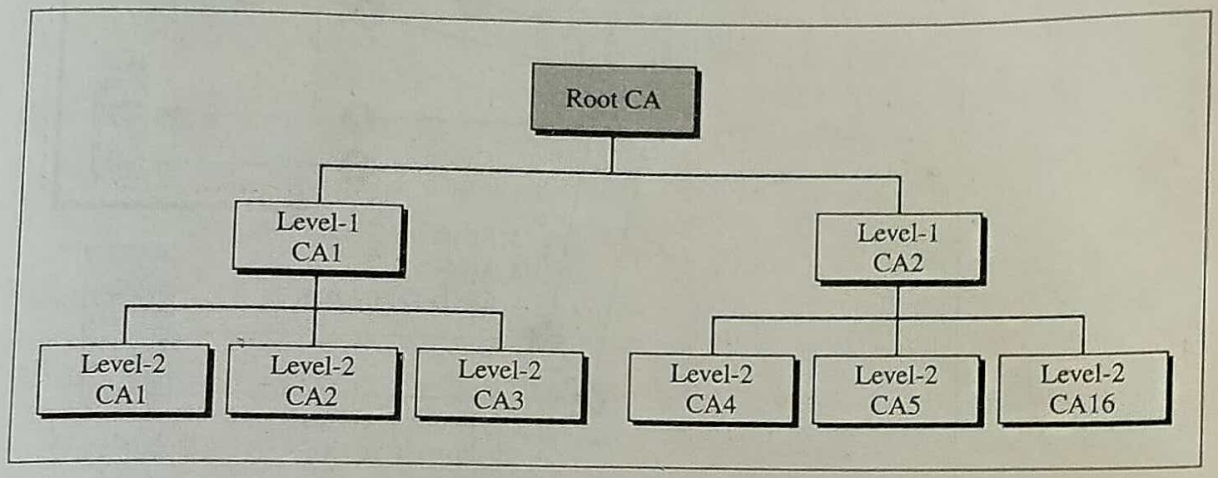
Step 4 The KDC sends an encrypted message to Alice that includes Alice's nonce, Bob's identity, the session key, and an encrypted ticket for Bob that includes his nonce. Now Alice has received the response to her nonce challenge and the session key.

Step 5 Alice sends Bob's ticket to him along with a new nonce, R_1, to challenge him.

may not have Bob's IP address. The local server can consult its parent server, up to the root, until the IP address is found.

Likewise, a solution to public-key queries is a hierarchical structure called a **public-key infrastructure (PKI).** Figure 30.16 shows an example of this hierarchy.

Figure 30.16 *PKI hierarchy*



At the first level, we can have a root CA that can certify the performance of CAs in the second level; these level-1 CAs may operate in a large geographic area or logical area. The level-2 CAs may operate in smaller geographic areas.

In this hierarchy, everybody trusts the root. But people may or may not trust intermediate CAs. If Alice needs to get Bob's certificate, she may find a CA somewhere to issue the certificate. But Alice may not trust that CA. In a hierarchy Alice can ask the next-higher CA to certify the original CA. The inquiry may go all the way to the root.

PKI is a new issue in the Internet. It will undoubtedly broaden in scope and change in the next few years.

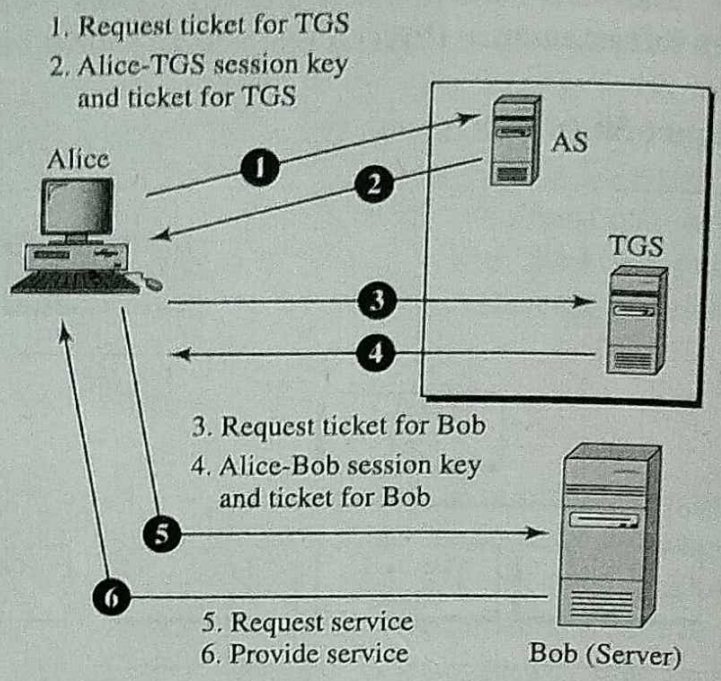
30.5 KERBEROS

Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular. Several systems including Windows 2000 use Kerberos. Kerberos is named after the three-headed dog in Greek mythology that guards the gates of Hades. Originally designed at MIT, it has gone through several versions. We discuss only version 4, the most popular, and we briefly explain the difference between version 4 and version 5, the latest.

Servers

Three servers are involved in the Kerberos protocol: an **authentication server (AS),** a **ticket-granting server (TGS),** and a real (data) server that provides services to others. In our examples and figures, *Bob* is the real server and *Alice* is the user requesting service. Figure 30.17 shows the relationship between these three servers.

Figure 30.17 *Kerberos servers*



Authentication Server (AS)

The AS is the KDC in the Kerberos protocol. Each user registers with the AS and is granted a user identity and a password. The AS has a database with these identities and the corresponding passwords. The AS verifies the user, issues a session key to be used between Alice and the TGS, and sends a ticket for the TGS.

Ticket-Granting Server (TGS)

The TGS issues a ticket for the real server (Bob). It also provides the session key (K_{AB}) between Alice and Bob. Kerberos has separated the user verification from ticket issuing. In this way, although Alice verifies her ID just once with AS, she can contact TGS multiple times to obtain tickets for different real servers.

Real Server

The real server (Bob) provides services for the user (Alice). Kerberos is designed for a client-server program such as FTP, in which a user uses the client process to access the server process. Kerberos is not used for person-to-person authentication.

Operation

A client process (Alice) can receive a service from a process running on the real server (Bob) in six steps, as shown in Figure 30.18.

Step 1

Alice sends her request to AS in plaintext, using her registered identity.

CHAPTER 31

Security Protocols in the Internet

All the security principles and concepts discussed in the previous two chapters can be used to provide all aspects of security for the Internet model. In particular, security measures can be applied to the network layer, transport layer, and application layer.

At the IP layer, implementation of security features is very complicated, especially since every device must be enabled. IP provides services not only for user applications, but also for other protocols such as OSPF, ICMP, and IGMP. This means that implementation of security at this level is not very effective unless all devices are equipped to use it. We discuss a protocol called IPSec that provides security at the IP level.

At the transport layer, security is even more complicated. We could modify the application or modify the transport layer for security. Instead, we discuss a protocol that "glues" a new layer to the transport layer to provide security on behalf of the transport layer.

At the application layer, each application is responsible for providing security. The implementation of security at this level is the simplest. It concerns two entities: the client and the server. We discuss a security method at the application layer called PGP.

A mechanism often used to ensure the integrity of an organization is a firewall. We give a brief discussion of firewalls in this chapter.

Finally, because this is the last chapter on security, we discuss an interesting technology—virtual private networks—that uses the public Internet but has the security level of a private network.

31.1 IP LEVEL SECURITY: IPSEC

IP Security (IPSec) is a collection of protocols designed by the IETF (Internet Engineering Task Force) to provide security for a packet at the IP level. IPSec does not define the use of any specific encryption or authentication method. Instead, it provides a framework and a mechanism; it leaves the selection of the encryption, authentication, and hashing methods to the user.

Security Association

IPSec requires a logical connection between two hosts using a *signaling protocol*, called **Security Association (SA).** In other words, IPSec needs the connectionless IP protocol changed to a connection-oriented protocol before security can be applied. An SA connection is a simplex (unidirectional) connection between a source and destination. If a duplex (bidirectional) connection is needed, two SA connections are required, one in each direction. An SA connection is uniquely defined by three elements:

1. A 32-bit security parameter index (SPI), which acts as a virtual circuit identifier in connection-oriented protocols such as Frame Relay or ATM.
2. The type of the protocol used for security. We will see shortly that IPSec defines two alternative protocols: AH and ESP.
3. The source IP address.

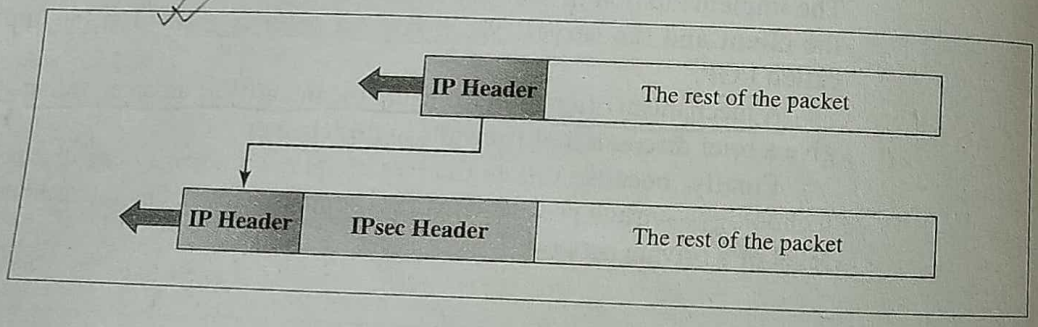
Two Modes

IPSec operates at two different modes: transport mode and tunnel mode. The mode defines where the IPSec header is added to the IP packet.

Transport Mode

In this mode, the IPSec header is added between the IP header and the rest of the packet, as shown in Figure 31.1.

Figure 31.1 *Transport mode*



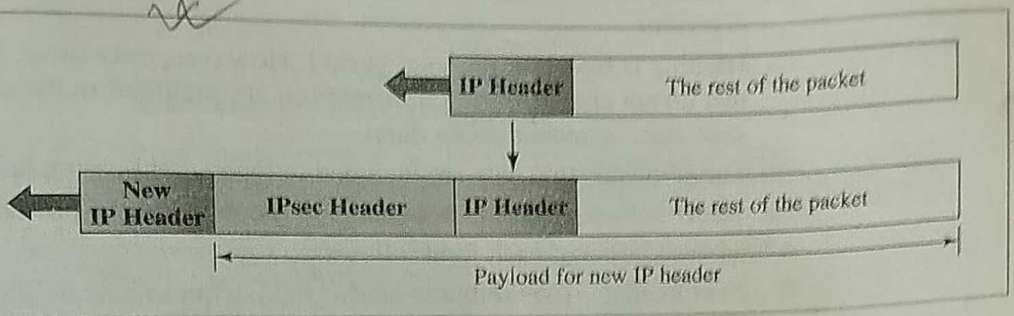
Tunnel Mode

In this mode, the IPSec header is placed in front of the original IP header. A new IP header is added in front. The IPSec header, the preserved IP header, and the rest of the packet are treated as the payload. Figure 31.2 shows the original and the new IP packet.

Two Security Protocols

IPSec defines two protocols: Authentication Header (AH) protocol and Encapsulating Security Payload (ESP) protocol. We discuss both of these protocols here.

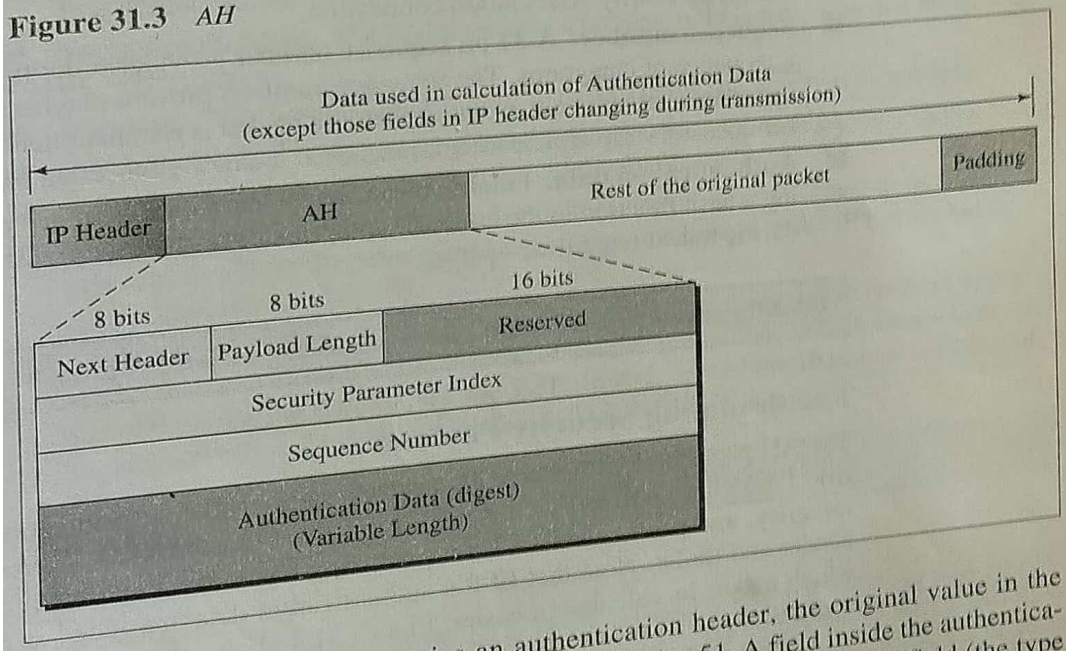
Figure 31.2 *Tunnel mode*



Authentication Header (AH) Protocol

The **Authentication Header (AH) protocol** is designed to authenticate the source host and to ensure the integrity of the payload carried by the IP packet. The protocol calculates a message digest, using a hashing function and a symmetric key, and inserts the digest in the authentication header. The AH is put in the appropriate location based on the mode (transport or tunnel). Figure 31.3 shows the fields and the position of the authentication header in the transport mode.

Figure 31.3 *AH*



When an IP datagram carries an authentication header, the original value in the protocol field of the IP header is replaced by the value 51. A field inside the authentication header (next header field) defines the original value of the protocol field (the type of payload being carried by the IP datagram). Addition of an authentication header follows these steps:

1. An authentication header is added to the payload with the authentication data field set to zero.

AES (Advanced Encr... n Standard) :-
AES is a widely used bl... iphen that encrypts
blocks of 128 bits using a ... of 128, 192 or 256 b

2. Padding may be added to make the total length even for a particular hashing algorithm.
3. Hashing is based on the total packet. However, only those fields of the IP header that do not change during transmission are included in the calculation of the message digest (authentication data).
4. The authentication data are included in the authentication header.
5. The IP header is added after changing the value of the protocol field to 51.

A brief description of each field follows:

- **Next header.** The 8-bit next-header field defines the type of payload carried by the IP datagram (TCP, UDP, ICMP, OSPF, and so on). It has the same function as the protocol field in the IP header before encapsulation. In other words, the process copies the value of the protocol field in the IP datagram to this field. The value of the protocol field in the IP datagram is changed to 51 to show that the packet carries an authentication header.
- **Payload length.** The name of this 8-bit payload-length field is misleading. It does not define the length of the payload; it defines the length of the authentication header in 4-byte multiples, but it does not include the first 8 bytes.
- **Security parameter index.** The 32-bit security parameter index (SPI) field plays the role of a virtual circuit identifier and is the same for all packets sent during a Security Association connection.
- **Sequence number.** A 32-bit sequence number provides ordering information for a sequence of datagrams. The sequence numbers prevent playback. Note that the sequence number is not repeated even if a packet is retransmitted. A sequence number does not wrap around after it reaches 2^{32}; a new connection must be established.
- **Authentication data.** Finally, the authentication data field is the result of applying a hash function to the entire IP datagram except for the fields that are changed during transit (e.g., time-to-live).

The AH protocol provides source authentication and data integrity, but not privacy.

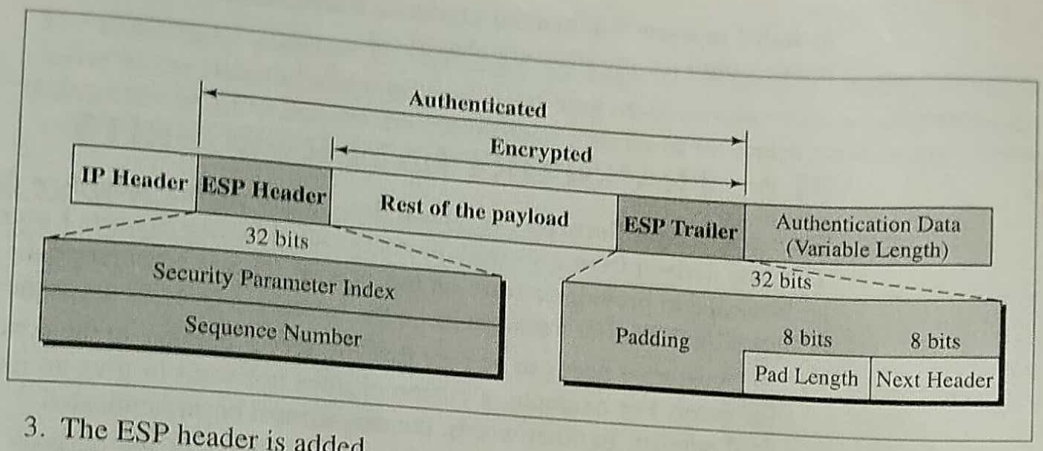
Encapsulating Security Payload

The AH protocol does not provide privacy, only source authentication and data integrity. IPSec later defined an alternative protocol that provides source authentication, integrity, and privacy called **Encapsulating Security Payload (ESP).** ESP adds a header and trailer. Note that ESP's authentication data are added at the end of packet which makes its calculation easier. Figure 31.4 shows the location of the ESP header and trailer.

When an IP datagram carries an ESP header and trailer, the value of the protocol field in the IP header changes to 50. A field inside the ESP trailer (the next-header field) holds the original value of the protocol field (the type of payload being carried by the IP datagram, such as TCP or UDP). The ESP procedure follows these steps:

1. An ESP trailer is added to the payload.
2. The payload and the trailer are encrypted.

Figure 31.4 *ESP*



3. The ESP header is added.
4. The ESP header, payload, and ESP trailer are used to create the authentication data.
5. The authentication data are added at the end of the ESP trailer.
6. The IP header is added after changing the protocol value to 50.

The fields for the header and trailer are as follows:

- **Security parameter index.** The 32-bit security parameter index field is similar to that defined for the AH protocol.
- **Sequence number.** The 32-bit sequence number field is similar to that defined for the AH protocol.
- **Padding.** This variable-length field (0 to 255 bytes) of 0s serves as padding.
- **Pad length.** The 8-bit pad length field defines the number of padding bytes. The value is between 0 and 255; the maximum value is rare.
- **Next header.** The 8-bit next-header field is similar to that defined in the AH protocol. It serves the same purpose as the protocol field in the IP header before encapsulation.
- **Authentication data.** Finally, the authentication data field is the result of applying an authentication scheme to parts of the datagram. Note the difference between the authentication data in AH and ESP. In AH, part of the IP header is included in the calculation of the authentication data; in ESP, it is not.

ESP provides source authentication, data integrity, and privacy.

IPv4 and IPv6

IPSec supports both IPv4 and IPv6. In IPv6, however, AH and ESP are part of the extension header.

AH versus ESP

The ESP protocol was designed after the AH protocol was already in use. ESP does whatever AH does with additional functionality (privacy). The question is, Why do we need AH? The answer is that we don't. However, the implementation of AH is already

included in some commercial products, which means that AH will remain part of the Internet until the products are phased out.

31.2 TRANSPORT LAYER SECURITY

Transport Layer Security (TLS) was designed to provide security at the transport layer. TLS was derived from a security protocol called Secure Sockets Layer (SSL), designed by Netscape to provide security on the WWW. TLS is a nonproprietary version of SSL designed by IETF. For transactions on the Internet, a browser needs the following:

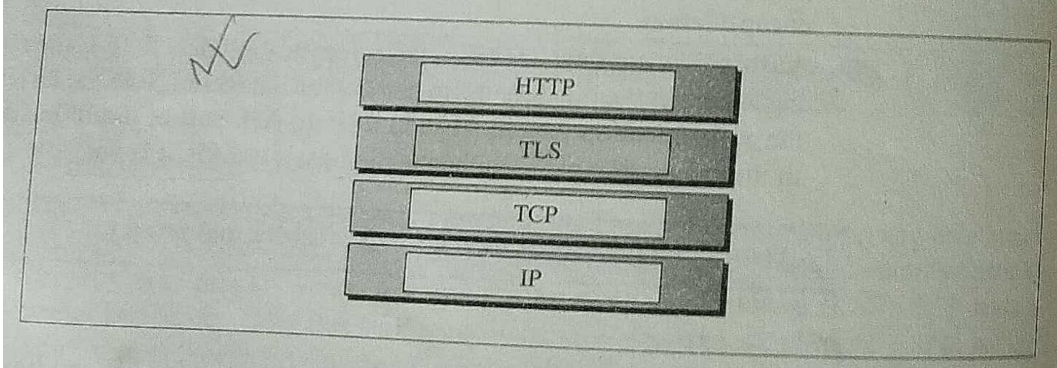
1. The customer needs to be sure that the server belongs to the actual vendor, not an imposter. For example, a customer does not want to give an imposter her credit card number. In other words, the server must be authenticated.
2. The customer needs to be sure that the contents of the message are not modified during transition. A bill for $100 must not be changed to $1000. The integrity of the message must be preserved.
3. The customer needs to be sure that an imposter does not intercept sensitive information such as a credit card number. There is a need for privacy.

There are other optional security aspects that can be added to the above list. For example, the vendor may need to authenticate the customer. TLS can provide additional features to cover these aspects of security.

Position of TLS

TLS lies between the application layer and the transport layer (TCP), as shown in Figure 31.5.

Figure 31.5 *Position of TLS*



The application layer protocol, in this case HTTP, uses the services of TLS, and TLS uses the services of the transport layer.

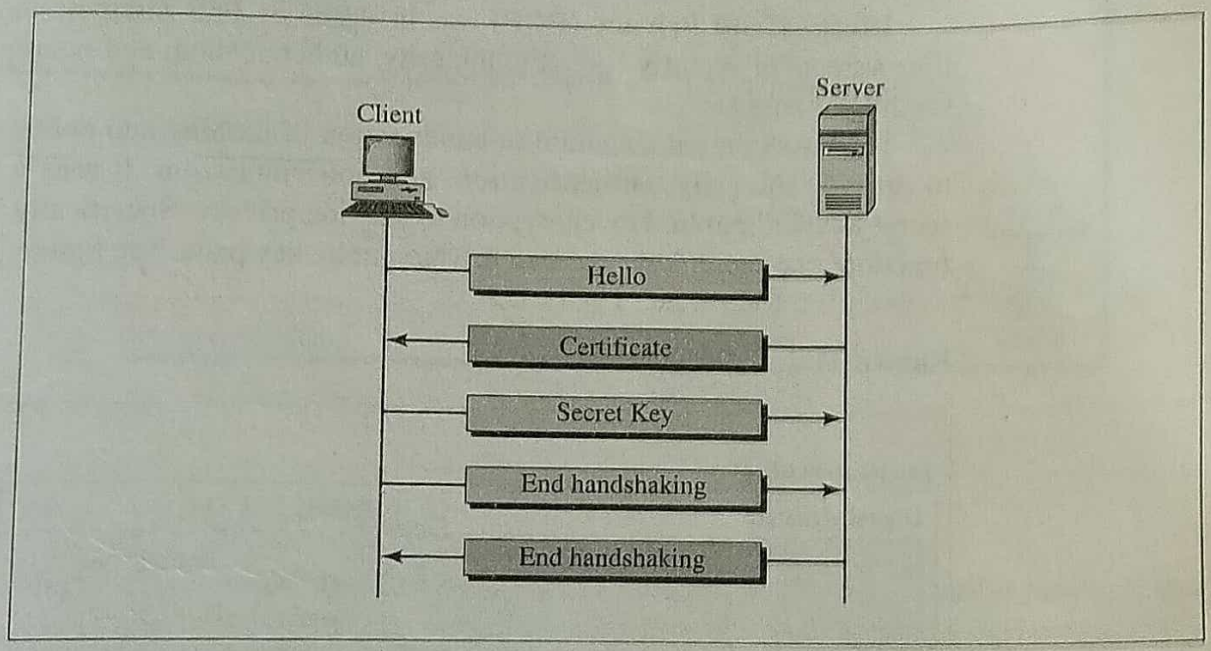
Two Protocols

TLS is actually two protocols: the handshake protocol and the data exchange (sometimes called the record) protocol.

Handshake Protocol

The **handshake protocol** is responsible for negotiating security, authenticating the server to the browser, and (optionally) defining other communication parameters. The handshake protocol defines the exchange of a series of messages between the browser and server. We discuss a simplified version, as shown in Figure 31.6.

Figure 31.6 *Handshake protocol*



1. The browser sends a *hello* message that includes the TLS version and some preferences.
2. The server sends a *certificate* message that includes the public key of the server. The public key is certified by some certification authority, which means that the public key is encrypted by a CA private key. The browser has a list of CAs and their public keys. It uses the corresponding key to decrypt the certificate and finds the server public key. This also authenticates the server because the public key is certified by the CA.
3. The browser generates a secret key, encrypts it with the server public key, and sends it to the server.
4. The browser sends a message, encrypted by the secret key, to inform the server that handshaking is terminating from the browser side.
5. The server decrypts the secret key using its private key and decrypts the message using the secret key. It then sends a message, encrypted by the secret key, to inform the browser that handshaking is terminating from the server side.

Note that handshaking uses the public key for two purposes: to authenticate the server and to encrypt the secret key, which is used in the data exchange protocol.

Data Exchange Protocol

The **data exchange** (record) **protocol** uses the secret key to encrypt the data for secrecy and to encrypt the message digest for integrity. The details and specification of algorithms are agreed upon during the handshake phase.

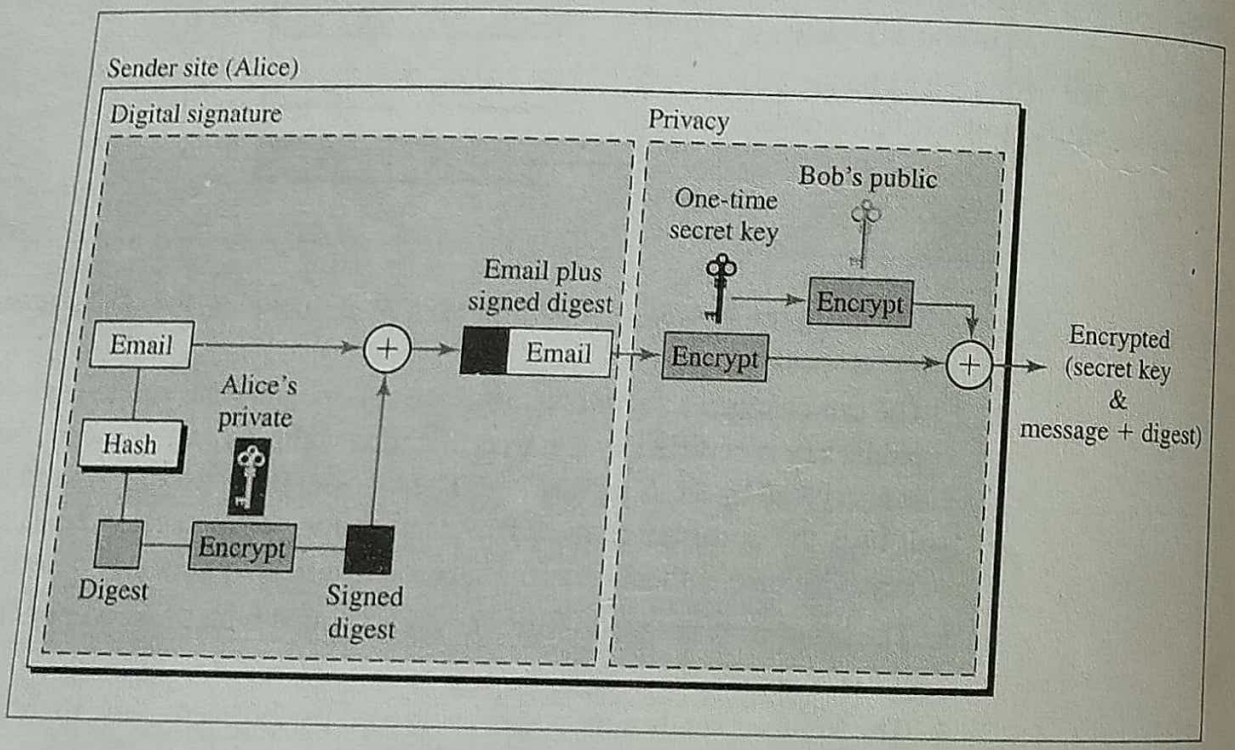
31.3 APPLICATION LAYER SECURITY: PGP

The implementation of security at the application layer is more feasible and simpler, particularly when the Internet communication involves only two parties, as in the case of email and TELNET. The sender and the receiver can agree to use the same protocol and to use any type of security services they desire. In this section, we discuss one protocol used at the application layer to provide security: PGP.

Pretty Good Privacy (PGP) was invented by Phil Zimmermann to provide all four aspects of security (privacy, integrity, authentication, and nonrepudiation) in the sending of email.

PGP uses digital signature (a combination of hashing and public-key encryption) to provide integrity, authentication, and nonrepudiation. It uses a combination of secret-key and public-key encryption to provide privacy. Specifically, it uses one hash function, one secret key, and two private-public key pairs. See Figure 31.7.

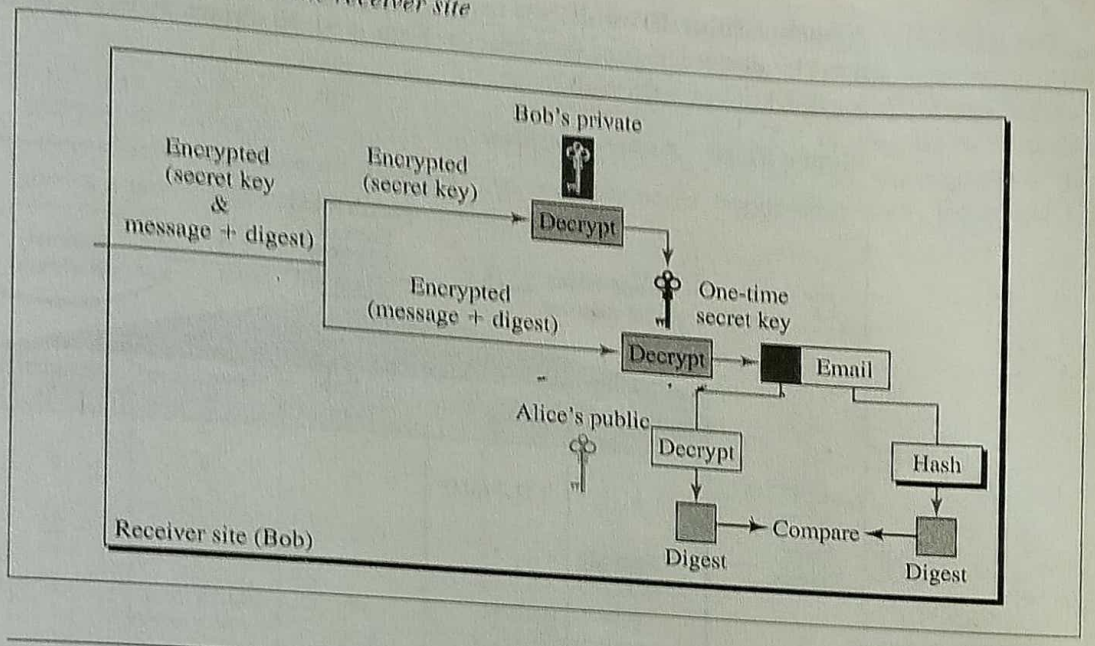
Figure 31.7 *PGP at the sender site*



The figure shows how PGP creates secure email at the sender site. The email message is hashed to create a digest. The digest is encrypted (signed) using Alice's private key. The message and the digest are encrypted using the one-time secret key created by Alice. The secret key is encrypted using Bob's public key and is sent together with the encrypted combination of message and digest.

Figure 31.8 shows how PGP uses hashing and a combination of three keys to extract the original message at the receiver site. The combination of encrypted secret key and message plus digest is received. The encrypted secret key first is decrypted (using Bob's private key) to get the one-time secret key created by Alice. The secret key then is used to decrypt the combination of the message plus digest.

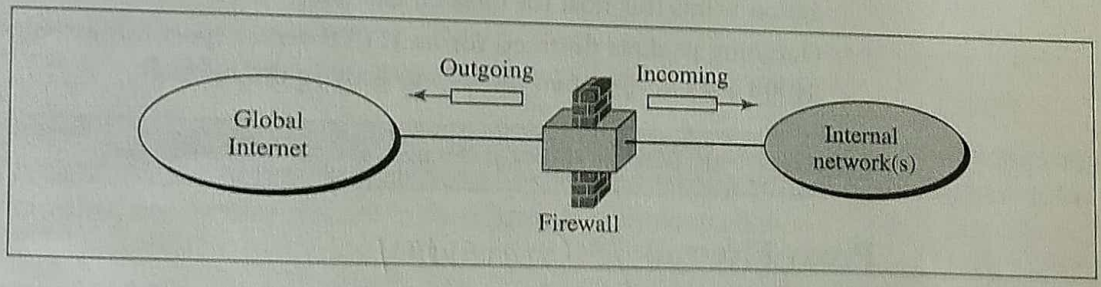
Figure 31.8 *PGP at the receiver site*



31.4 FIREWALLS

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 31.9 shows a firewall.

Figure 31.9 *Firewall*



For example, a firewall may filter all incoming packets destined for a specific host or a specific server such as HTTP. A firewall can be used to deny access to a specific host or a specific service in the organization.

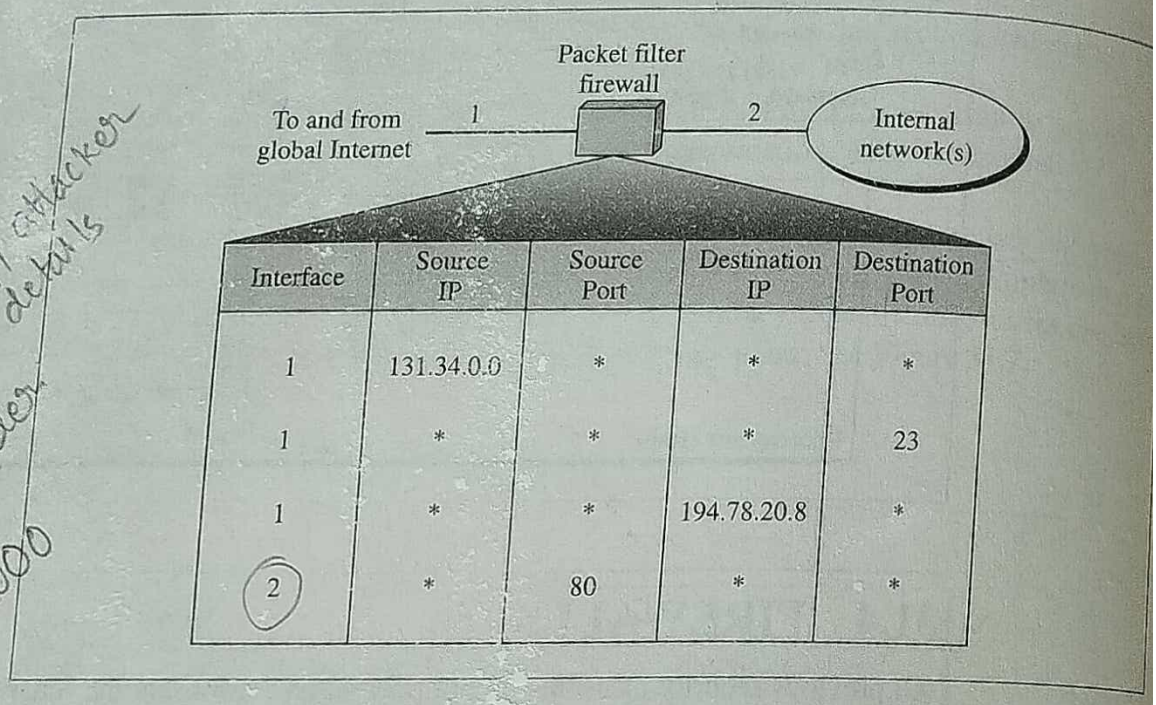
A firewall is usually classified as a packet-filter firewall or a proxy-based firewall.

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 pac must be discarded (not forwarded). Figure 31.10 shows an example of a filtering ta for this kind of a firewall.

Figure 31.10 *Packet-filter firewall*



According to the figure, the following packets are filtered:

1. Incoming packets from network 131.34.0.0. are blocked (security precaution). Note that the * (asterisk) means "any."
2. Incoming packets destined for any internal TELNET server (port 23) are blocked.
3. Incoming packets destined for internal host 194.78.20.8. are blocked. The organization wants this host for internal use only.
4. Outgoing packets destined for an HTTP server (port 80) are blocked. The organization does not want employees to browse the Internet.

A packet-filter firewall filters at the network or transport layer.

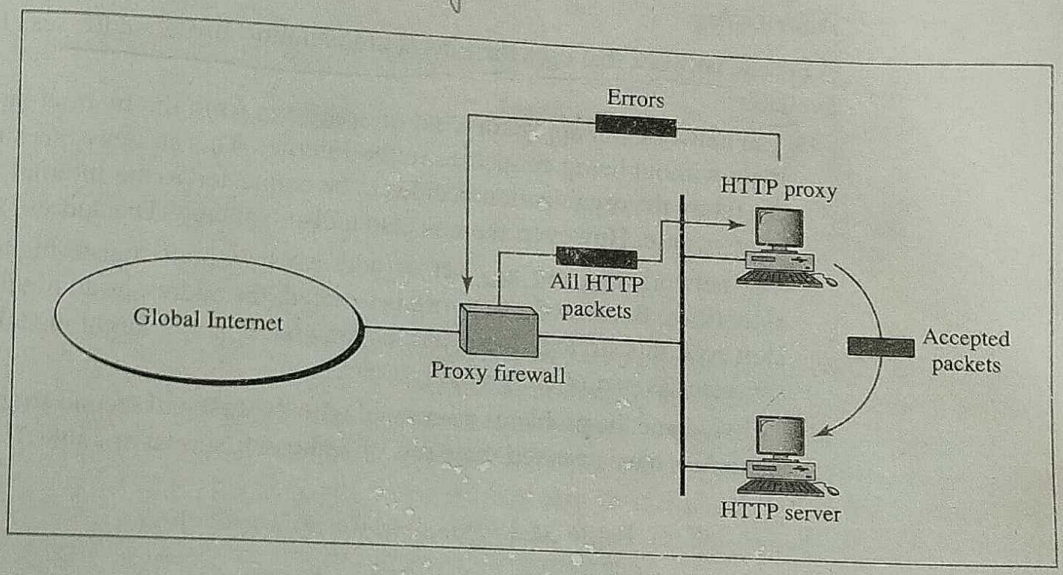
Proxy Firewall / Gateway

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). As an example, assume that an organization wants to implement the following policies regarding its Web pages: Only those Internet users who have previously established business relations with the company can have access; access to other users must be blocked. In this case, a packet-filter firewall is not feasible because it cannot distinguish between different packets arriving at TCP port 80 (HTTP). Testing must be done at the application level (using URLs).

One solution is to install a proxy computer (sometimes called an application gateway), which stands between the customer (user client) computer and the corporation

computer. When the user client process sends a message, the **proxy firewall** run 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. In this way, the requests of the external users are filtered based on the contents at the application layer. Figure 31.11 shows a proxy firewall implementation.

Figure 31.11 *Proxy firewall*



A proxy firewall filters at the application layer.

31.5 VIRTUAL PRIVATE NETWORK

Virtual private network (VPN) is a technology that is gaining popularity among large organizations that use the global Internet for both intra- and interorganization communication, but require privacy in their internal communication.

Private Networks

A private network is designed for use inside an organization. It allows access to shared resources and, at the same time, provides privacy. Before we discuss some aspects of these networks, let us define two commonly used related terms: *intranet* and *extranet*.

Intranet

An **intranet** is a private network (LAN) that uses the Internet model. However, access to the network is limited to the users inside the organization. The network uses application programs defined for the global Internet, such as HTTP, and may have Web servers, print servers, file servers, and so on.

Extranet

An **extranet** is the same as an intranet with one major difference: Some resources may be accessed by specific groups of users outside the organization under the control of the network administrator. For example, an organization may allow authorized customers access to product specifications, availability, and online ordering. A university or a college can allow distance learning students access to the computer lab after passwords have been checked.

Addressing

A private network that uses the Internet model must use IP addresses. Three choices are available:

1. The network can apply for a set of addresses from the Internet authorities and use them without being connected to the Internet. This strategy has an advantage. If in the future the organization decides to be connected to the Internet, it can do so with relative ease. However, there is also a disadvantage: The address space is wasted.
2. The network can use any set of addresses without registering with the Internet authorities. Because the network is isolated, the addresses do not have to be unique. However, this strategy has a serious drawback: Users might mistakenly confuse the addresses as part of the global Internet.
3. To overcome the problems associated with the first and second strategies, the Internet authorities have reserved three sets of addresses, shown in Table 31.1.

Table 31.1 *Addresses for private networks*

Prefix	*Range*	*Total*
10/8	10.0.0.0 to 10.255.255.255	2^{24}
172.16/12	172.16.0.0 to 172.31.255.255	2^{20}
192.168/16	192.168.0.0 to 192.168.255.255	2^{16}

Any organization can use an address out of this set without permission from the Internet authorities. Everybody knows that these reserved addresses are for private networks. They are unique inside the organization, but they are not unique globally. No router will forward a packet that has one of these addresses as the destination address.

Achieving Privacy

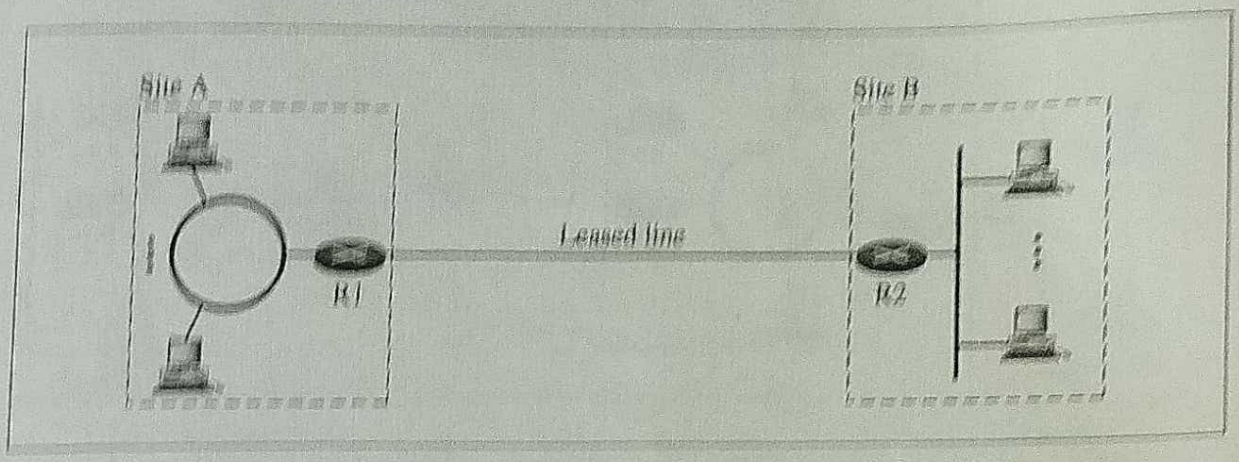
To achieve privacy, organizations can use one of three strategies: private networks, hybrid networks, and virtual private networks.

Private Networks

An organization that needs privacy when routing information inside the organization can use a **private network** as discussed previously. A small organization with one single site can use an isolated LAN. People inside the organization can send data to one another that totally remain inside the organization, secure from outsiders. A larger organization with

several sites can create a private internet. The LANs at different sites can be connected to each other using routers and leased lines. In other words, an internet can be made out of private LANs and private WANs. Figure 31.12 shows such a situation for an organization with two sites. The LANs are connected to each other using routers and one leased line.

Figure 31.12 *Private network*



In this situation, the organization has created a private internet that is totally isolated from the global Internet. For end-to-end communication between stations at different sites, the organization can use the Internet model. However, there is no need for the organization to apply for IP addresses with the Internet authorities. It can use private IP addresses. The organization can use any IP class and assign network and host addresses internally. Because the internet is private, duplication of addresses by another organization in the global Internet is not a problem.

Hybrid Networks

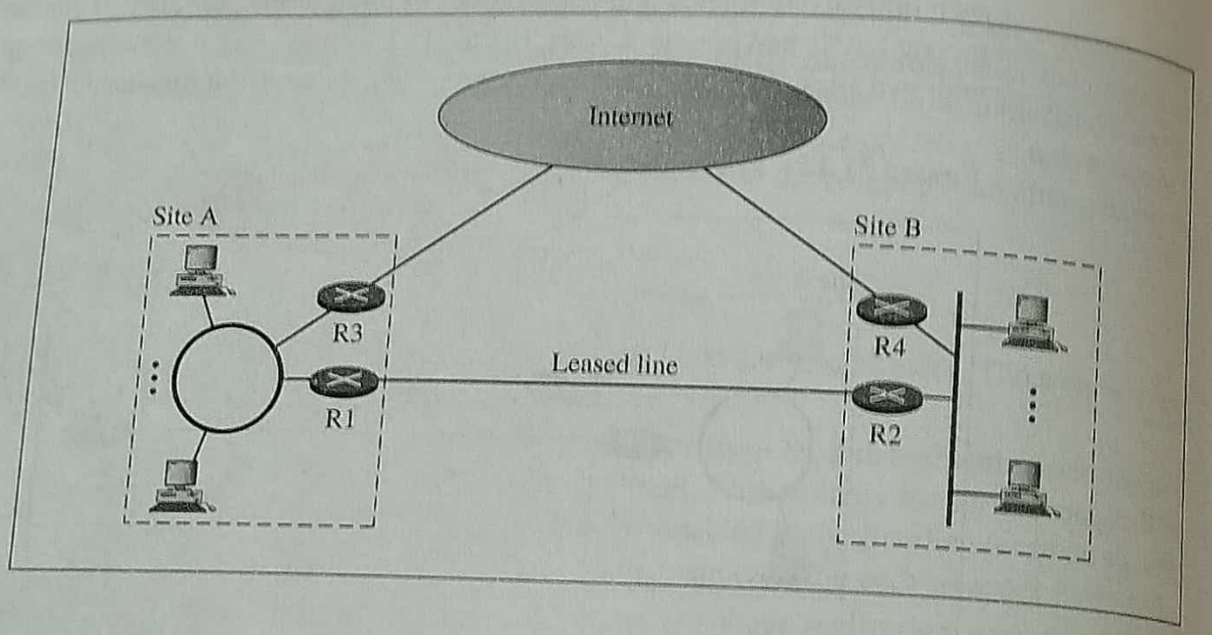
Today, most organizations need to have privacy in intraorganization data exchange, but, at the same time, they need to be connected to the global Internet for data exchange with other organizations. One solution is the use of a **hybrid network.** A hybrid network allows an organization to have its own private internet and, at the same time, access to the global Internet. Intraorganization data are routed through the private internet; interorganization data are routed through the global Internet. Figure 31.13 shows an example of this situation.

An organization with two sites uses routers R1 and R2 to connect the two sites privately through a leased line; it uses routers R3 and R4 to connect the two sites to the rest of the world. The organization uses global IP addresses for both types of communication. However, packets destined for internal recipients are routed only through routers R1 and R2. Routers R3 and R4 route the packets destined for outsiders.

Virtual Private Networks

Both private and hybrid networks have a major drawback: cost. Private wide-area networks (WANs) are expensive. To connect several sites, an organization needs several leased lines, which means a high monthly fee. One solution is to use the global Internet for both private and public communications. A technology called virtual private network (VPN) allows organizations to use the global Internet for both purposes.

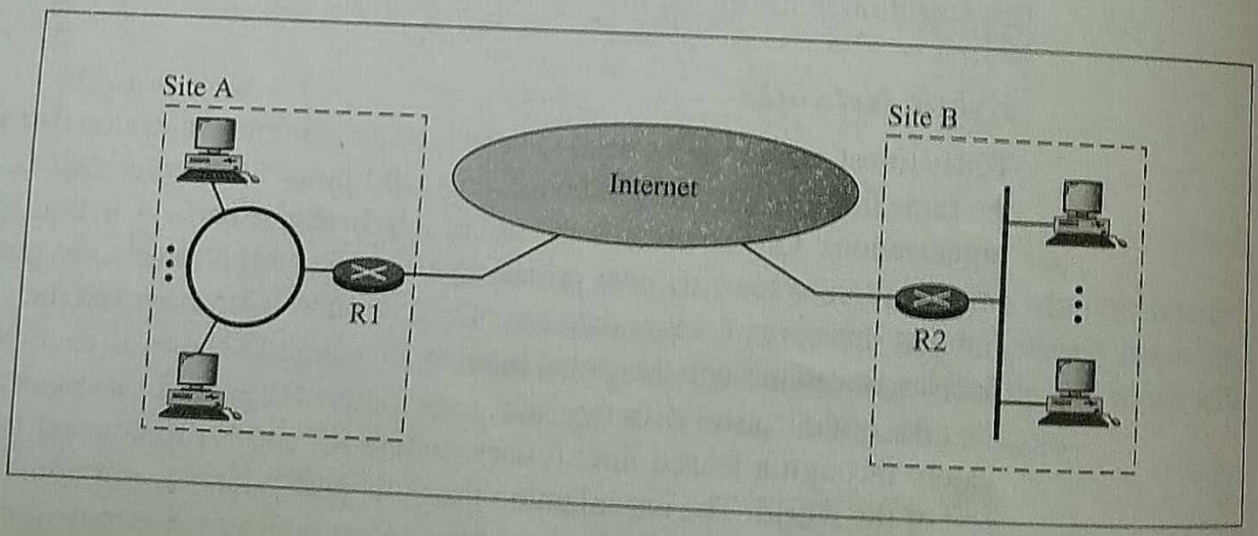
Figure 31.13 *Hybrid network*



VPN creates a network that is private but virtual. It is private because it guarantees privacy inside the organization. It is virtual because it does not use real private WANs; the network is physically public but virtually private.

Figure 31.14 shows the idea of a virtual private network. Routers R1 and R2 use VPN technology to guarantee privacy for the organization.

Figure 31.14 *Virtual private network*



VPN Technology

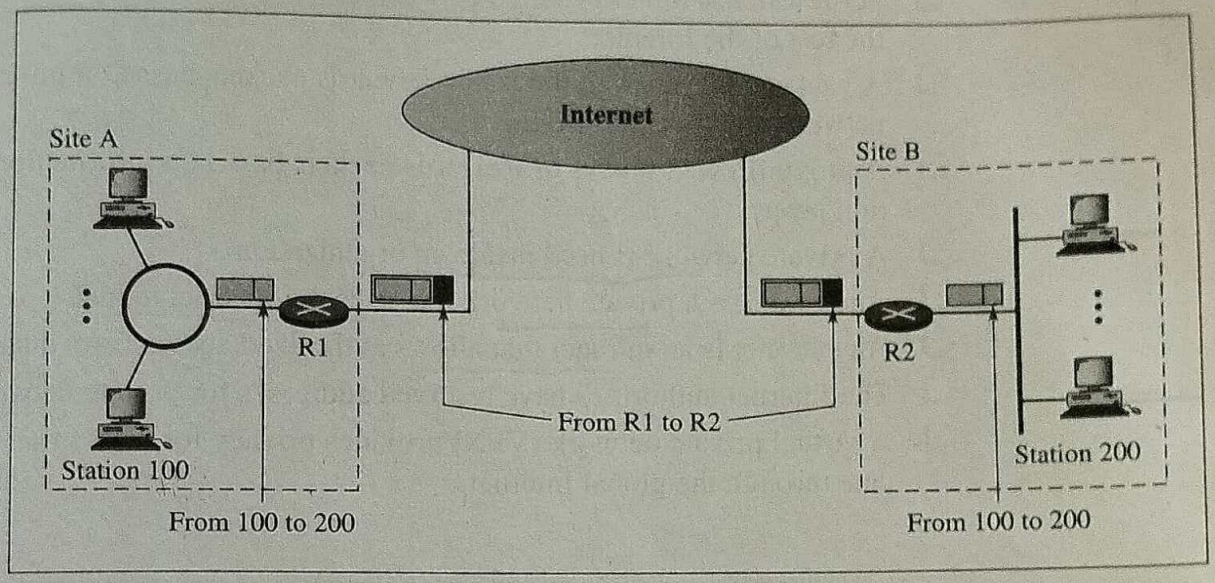
VPN technology uses IPSec in the tunnel mode to provide authentication, integrity, and privacy.

Tunneling

To guarantee privacy and other security measures for an organization, VPN can use the IPSec in the tunnel mode. In this mode, each IP datagram destined for private use in the

organization is encapsulated in another datagram. To use IPSec in the **tunneling** mode, the VPNs need to use two sets of addressing, as shown in Figure 31.15.

Figure 31.15 *Addressing in a VPN*



The public network (Internet) is responsible for carrying the packet from R1 to R2. Outsiders cannot decipher the contents of the packet or the source and destination addresses. Deciphering takes place at R2, which finds the destination address of the packet and delivers it.

31.6 KEY TERMS

Authentication Header (AH) protocol
data exchange protocol
Encapsulating Security Payload (ESP)
extranet
firewall
handshake protocol
hybrid network
intranet
IP Security (IPSec)
packet-filter firewall
Pretty Good Privacy (PGP)
private network
proxy firewall
Security Association (SA)
Transport Layer Security (TLS)
tunneling
virtual private network (VPN)

31.7 SUMMARY

- Security methods can be applied in the application layer, transport layer, and IP layer.
- IP Security (IPSec) is a collection of protocols designed by the IETF to provide security for an Internet packet.
- The Authentication Header protocol provides integrity and message authentication.
- The Encapsulating Security Payload protocol provides integrity, message authentication, and privacy.

- ❑ Transport Layer Security (TLS) provides security at the transport layer through its handshake protocol and data exchange protocol.
- ❑ Pretty Good Privacy (PGP) provides security for the transmission of email.
- ❑ A firewall is a router installed between the internal network of an organization and the rest of the Internet.
- ❑ A packet-filter firewall blocks or forwards packets based on information in the network and transport layers.
- ❑ A proxy firewall blocks or forwards packets based on information in the application layer.
- ❑ A private network is used inside an organization.
- ❑ An intranet is a private network that uses the Internet model.
- ❑ An extranet is an intranet that allows authorized access from outside users.
- ❑ The Internet authorities have reserved addresses for private networks.
- ❑ A virtual private network (VPN) provides privacy for LANs that must communicate through the global Internet.

31.8 PRACTICE SET

Review Questions

1. What is IPSec?
2. Why does IPSec need Security Association?
3. What are the two protocols defined by IPSec?
4. What does AH add to the IP packet?
5. What does ESP add to the IP packet?
6. What are authentication data?
7. What is the security parameter index field?
8. Are both AH and ESP needed for IP security? Why or why not?
9. What are the two protocols defined by TLS?
10. What is the name of the protocol that provides security for email?
11. What is the purpose of a firewall?
12. What are the two types of firewalls?
13. What is a VPN and why is it needed?
14. How do LANs on a fully private internet communicate?
15. How do LANs on a hybrid internet communicate?

Multiple-Choice Questions

16. IPSec requires a logical connection between two hosts using a signaling protocol called _____.
    a. AH
    b. SA

c. PGP
d. TLS

17. The handshake protocol and data exchange protocol are part of _____.
a. CA
b. KDC
c. TLS
d. SSH

18. _____ is a collection of protocols that provide security at the IP layer level.
a. TLS
b. SSH
c. PGP
d. IPSec

19. _____ is an IP layer security protocol that only provides integrity and authentication.
a. AH
b. PGP
c. ESP
d. IPSec

20. _____ is an IP layer security protocol that provides privacy as well as integrity and authentication.
a. AH
b. PGP
c. ESP
d. IPSec

21. An IP datagram carries an authentication header if the _____ field of the IP header has a value of 51.
a. Next-header
b. Protocol
c. Security parameter index
d. Sequence number

22. A _____ can forward or block packets based on the information in the network layer and transport layer headers.
a. Proxy firewall
b. Packet-filter firewall
c. Message digest
d. Private key

23. The _____ field in the authentication header and the ESP header define the security method used in creating the authentication data.
a. Padding
b. Sequence number
c. Authentication data
d. SPI