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.

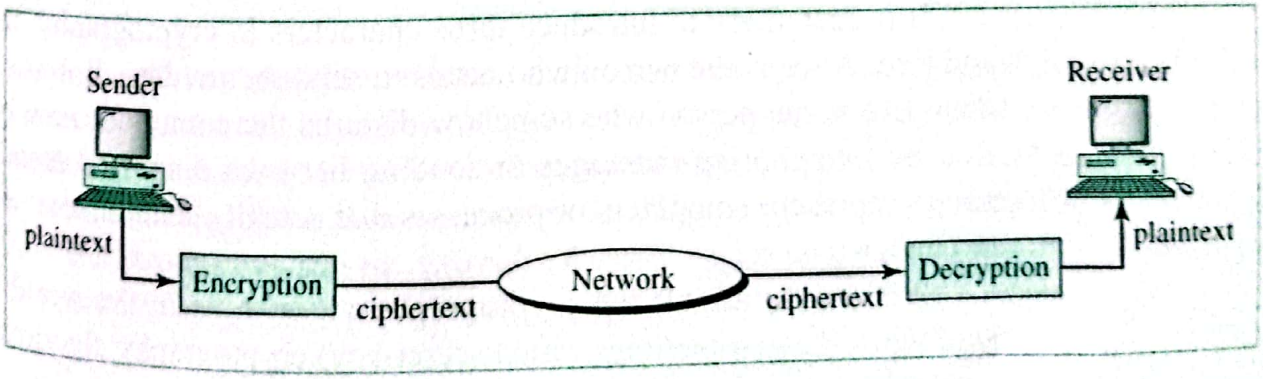
We focus on symmetric-key cryptography, which is presently more common than public-key cryptography. Symmetric-key cryptography is less math-based than public-key cryptography, which has its origins in number theory.

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*



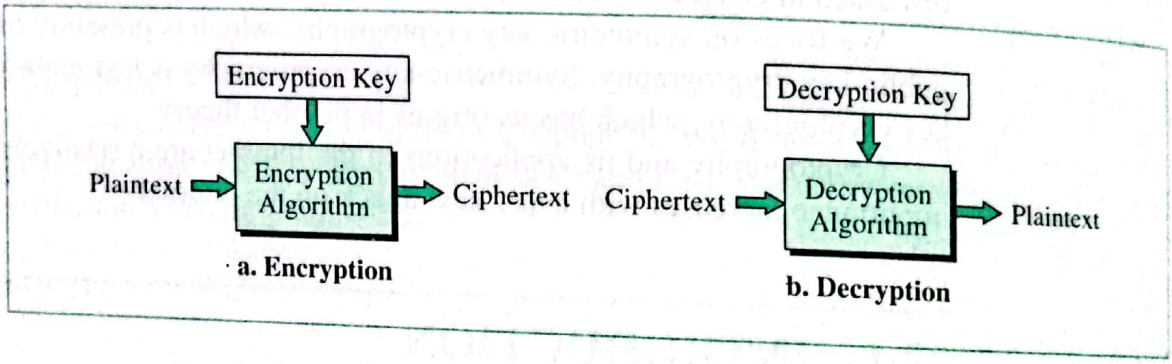
The original message, before being transformed, is called **plaintext.** After the message is transformed, it is called **ciphertext.** An **encryption** algorithm transforms the plaintext to ciphertext; a **decryption** algorithm transforms the ciphertext back to

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.

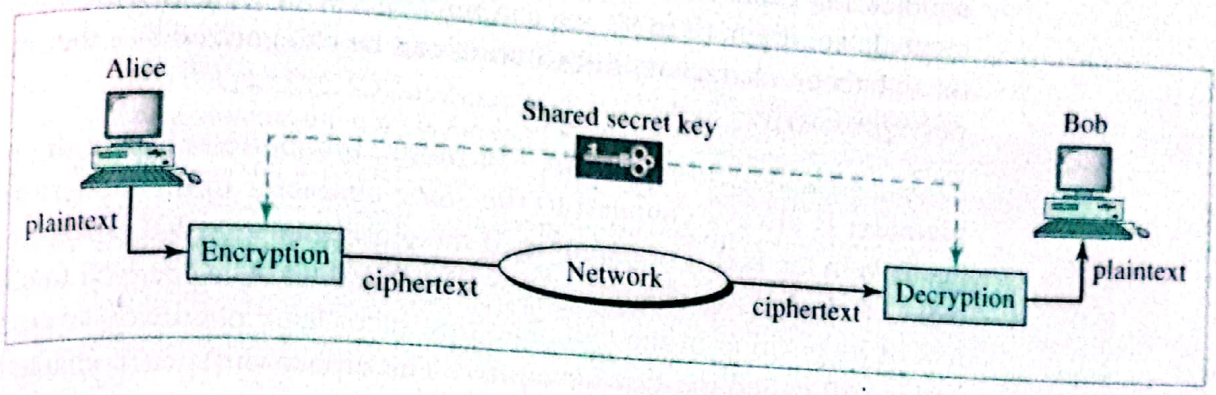
It is customary to introduce three characters in cryptography; we use Alice, Bob, and Eve. Alice is the person who needs to send secure data. Bob is the recipient of the data. Eve is the person who somehow disturbs the communication between Alice and Bob by intercepting messages or sending her own disguised 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.

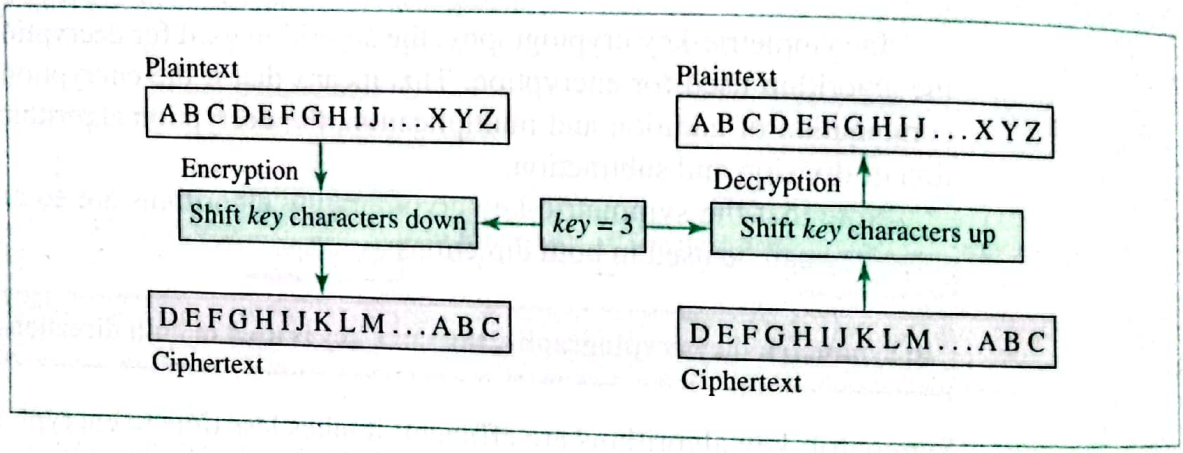
Substitution Cipher

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*



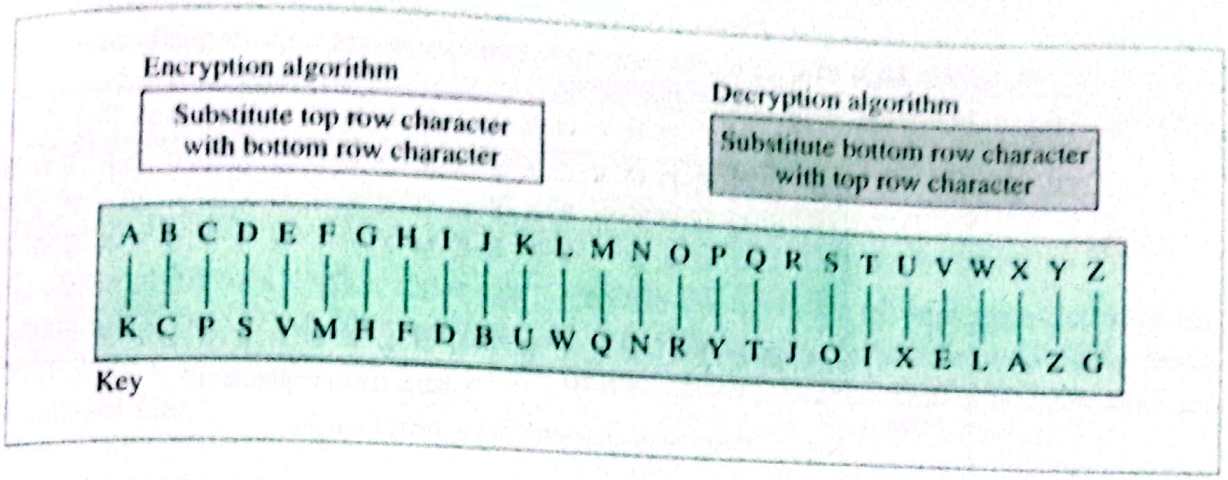
Before we go further, let us analyze the Caesar cipher which has an encryption algorithm, a decryption algorithm, and a symmetric key. As the figure shows, the encryption algorithm is "shift *key* characters down." The decryption algorithm is "shift *key* characters up." The key is 3. Note that the encryption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

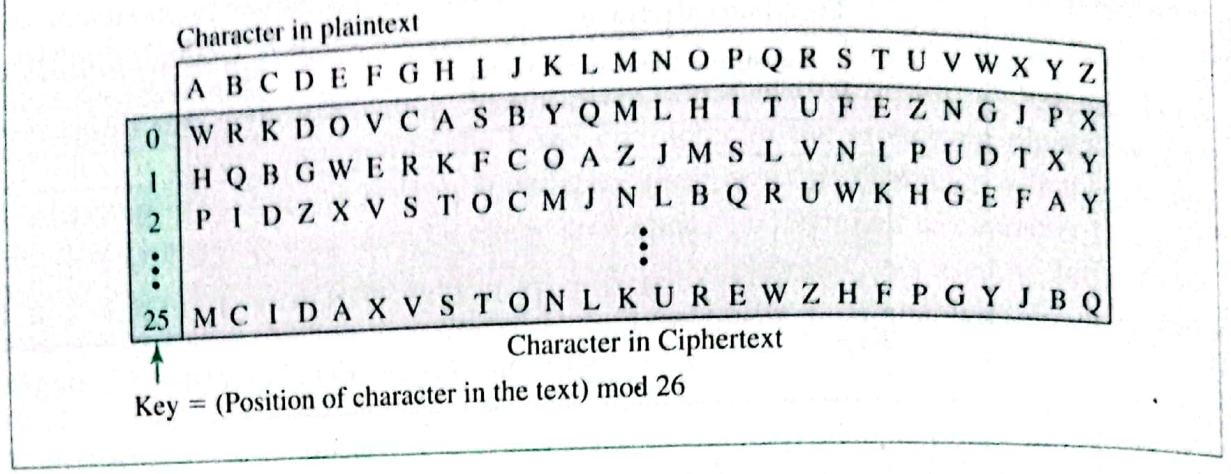
language being used. For example, in English, the most frequently used characters are E, T, O, and A. An attacker can easily break the code by finding which character is used the most and replace that one with the letter E. It can then find the next most frequent and replace it with T, and so on.

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*



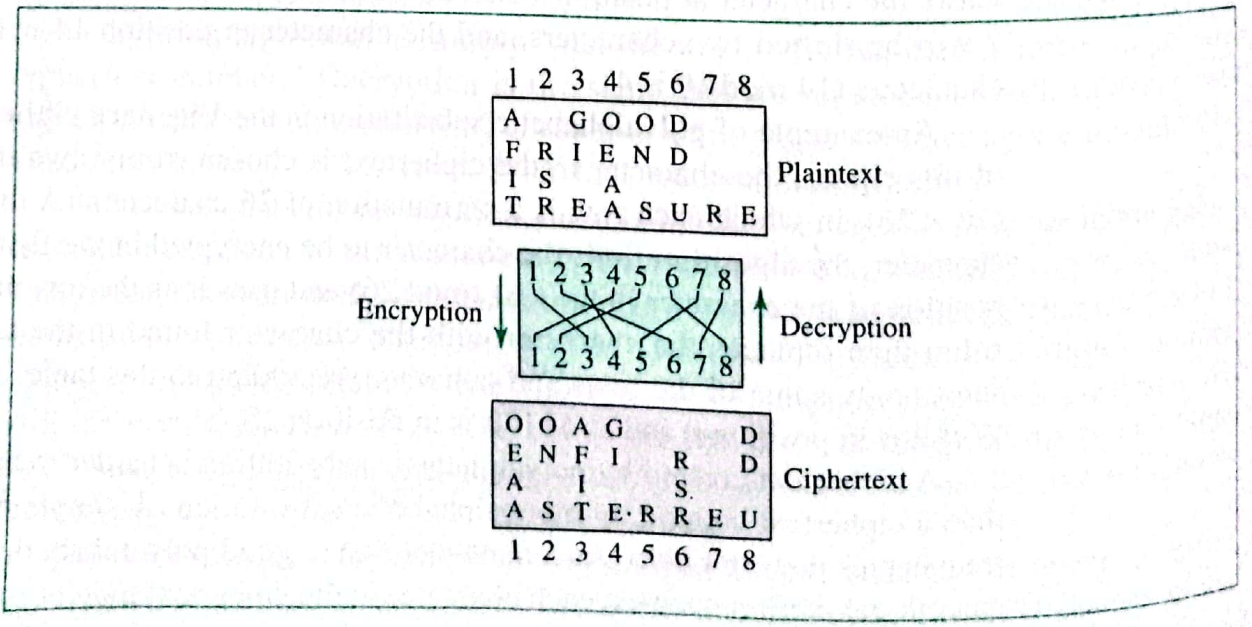
In polyalphabetic substitution, the relationship between a character in the plaintext and a character in the ciphertext is one-to-many.

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*

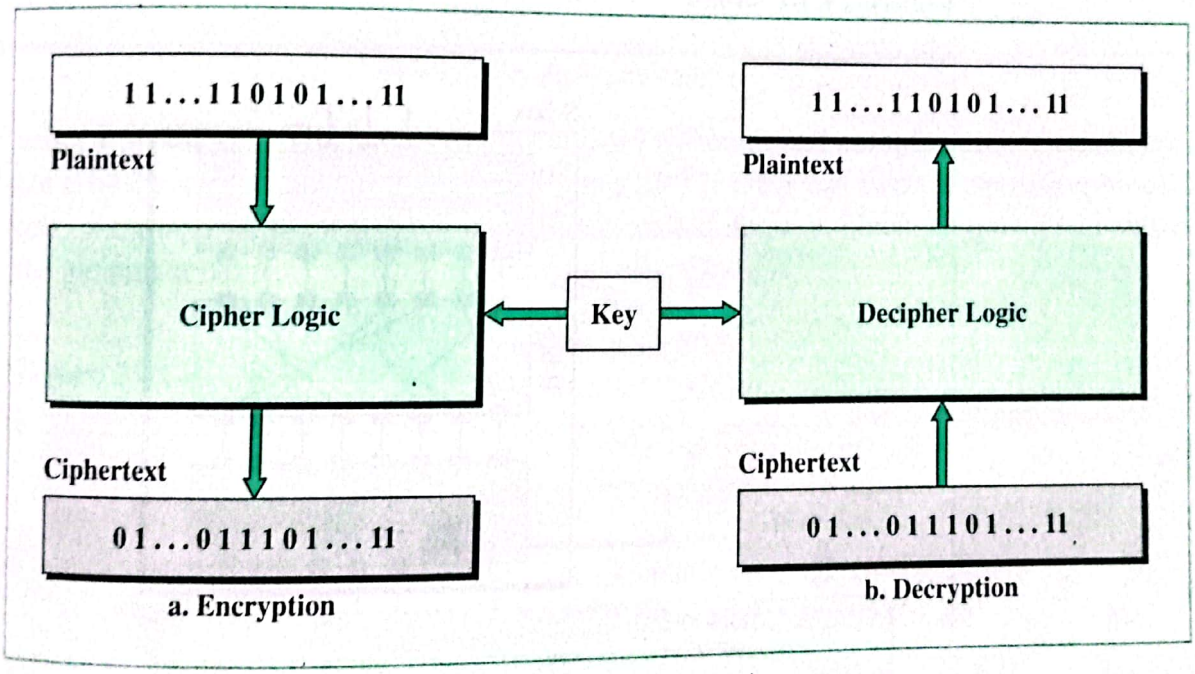


of transpositional cryptography. The key defines which columns should be swapped. As you have guessed, transpositional cryptography is not very secure either. The character frequencies are preserved, and the attacker can find the plaintext through trial and error. This method can be combined with other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

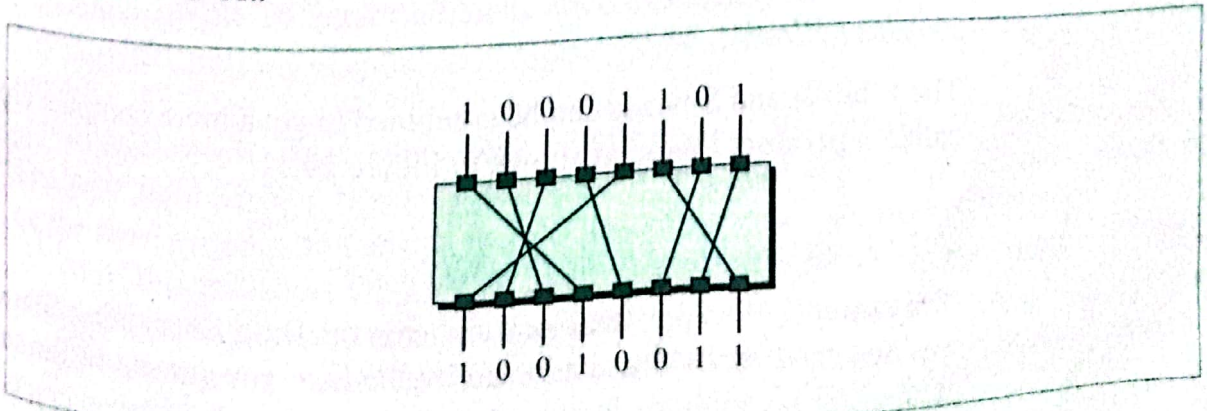
Figure 29.8 *Block cipher*



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*

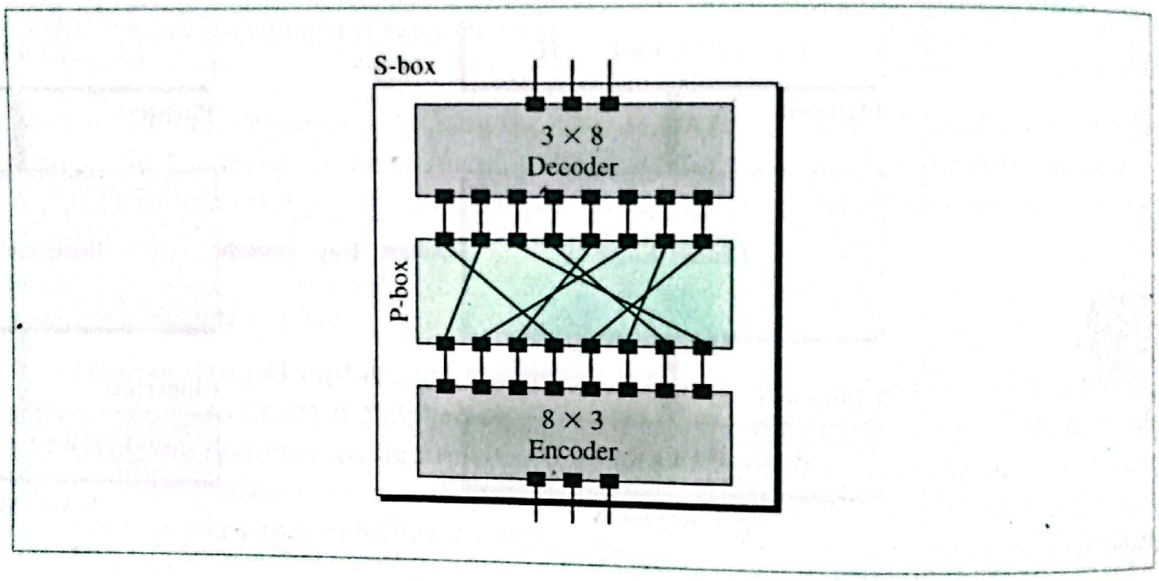


is faster. The key and the encryption/decryption algorithm are normally embedded in the hardware. Note that both the plaintext and ciphertext have the same number of 1s and 0s.

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.

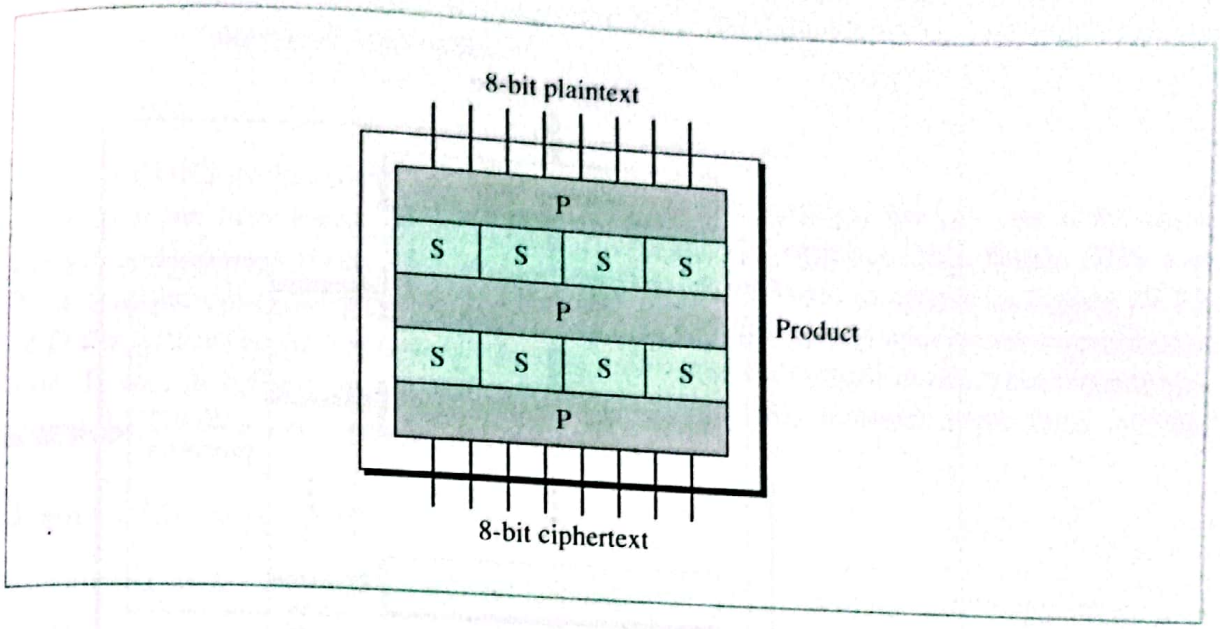
Product Block

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)

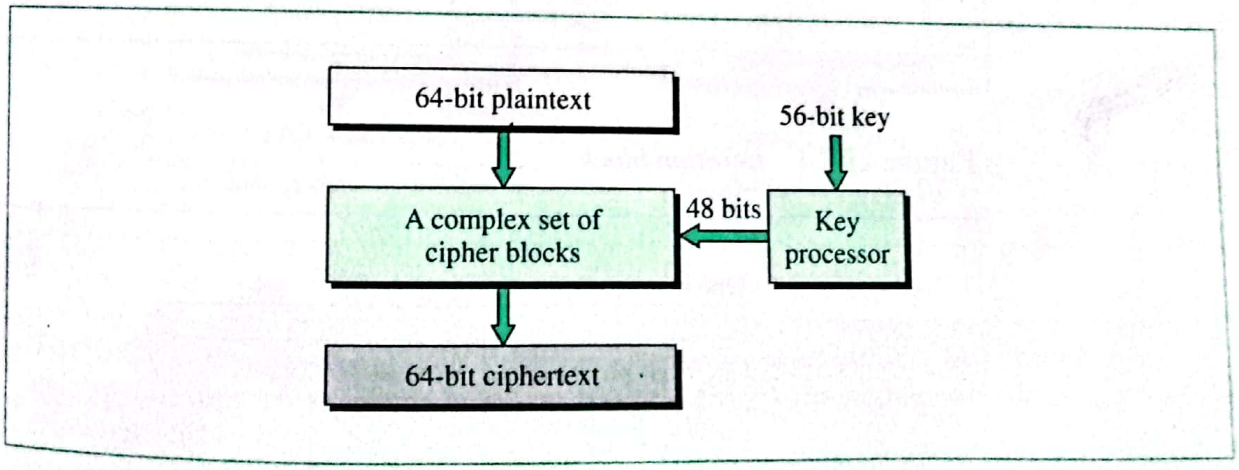
One example of a complex block cipher is the **Data Encryption Standard (DES).** DES was designed by IBM and adopted by the U.S. government as the standard encryption method for nonmilitary and nonclassified use. The algorithm encrypts a 64-bit plaintext

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*

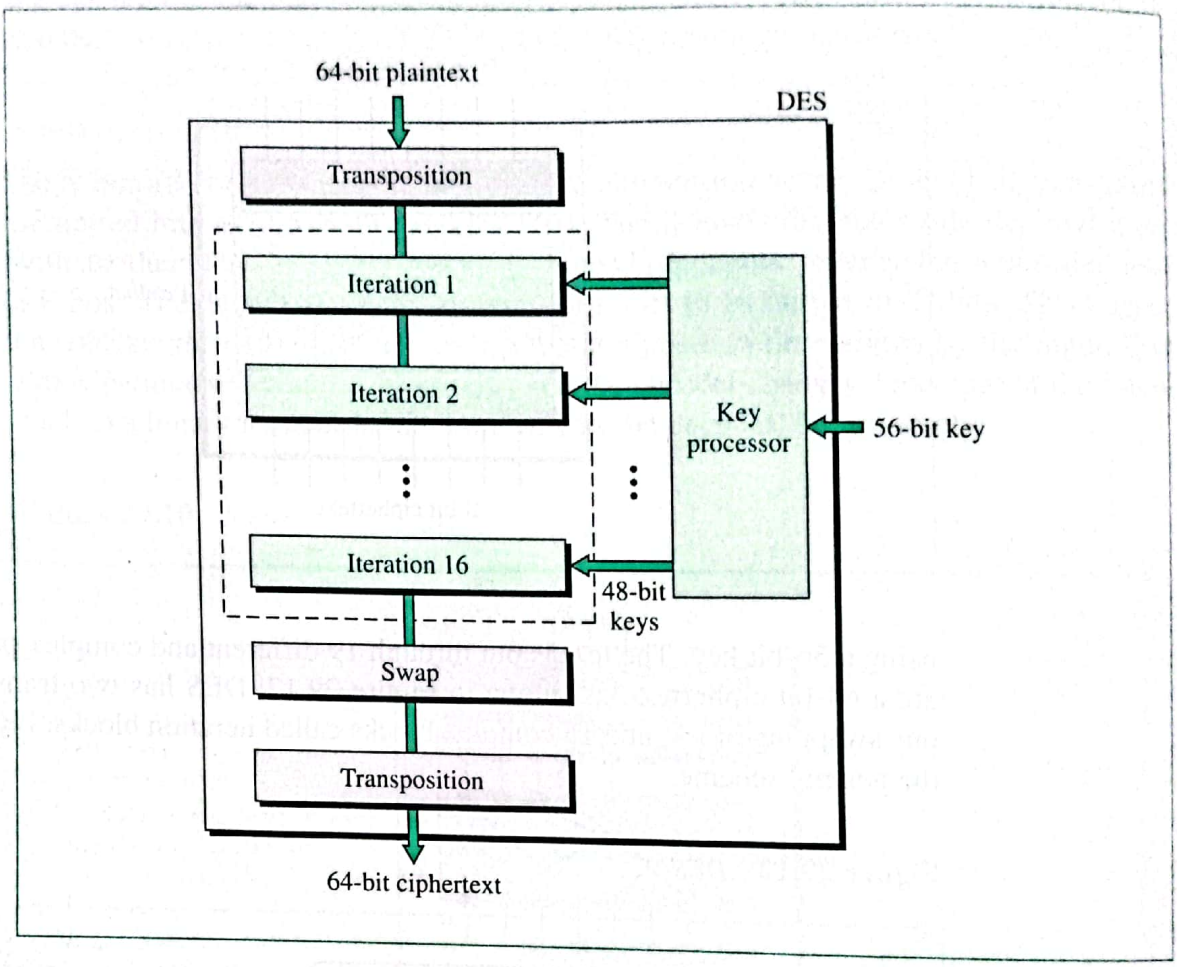
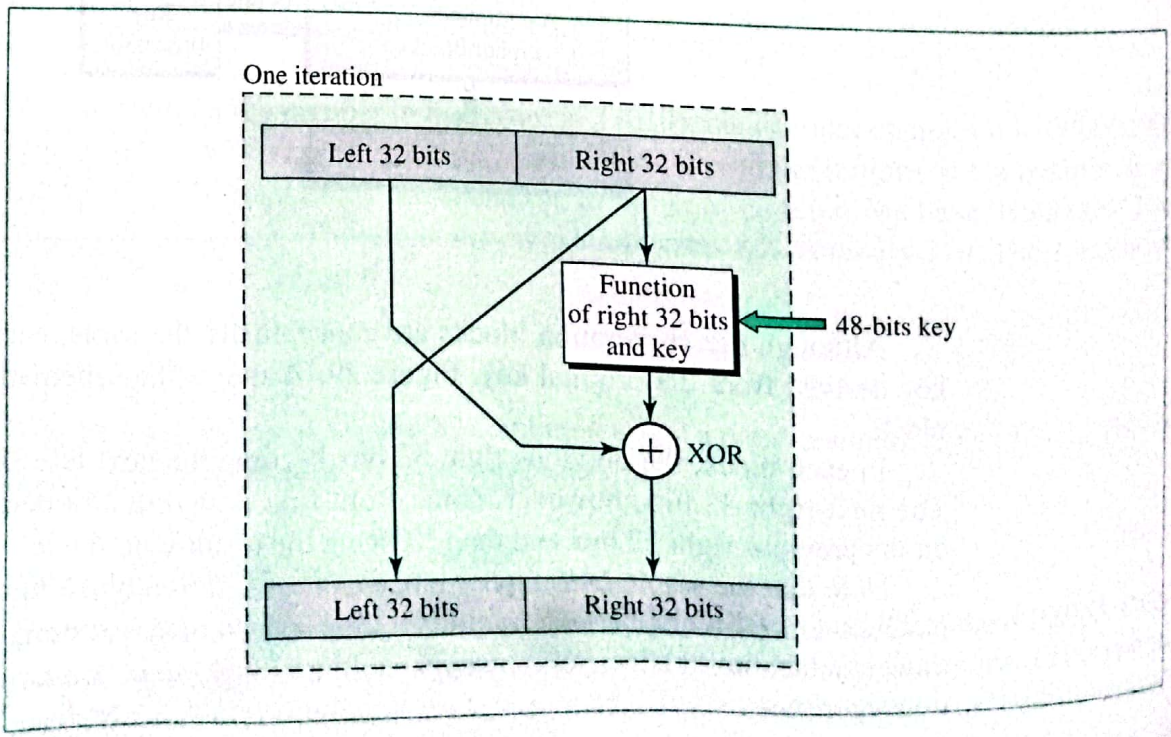


Figure 29.14 *Iteration block*

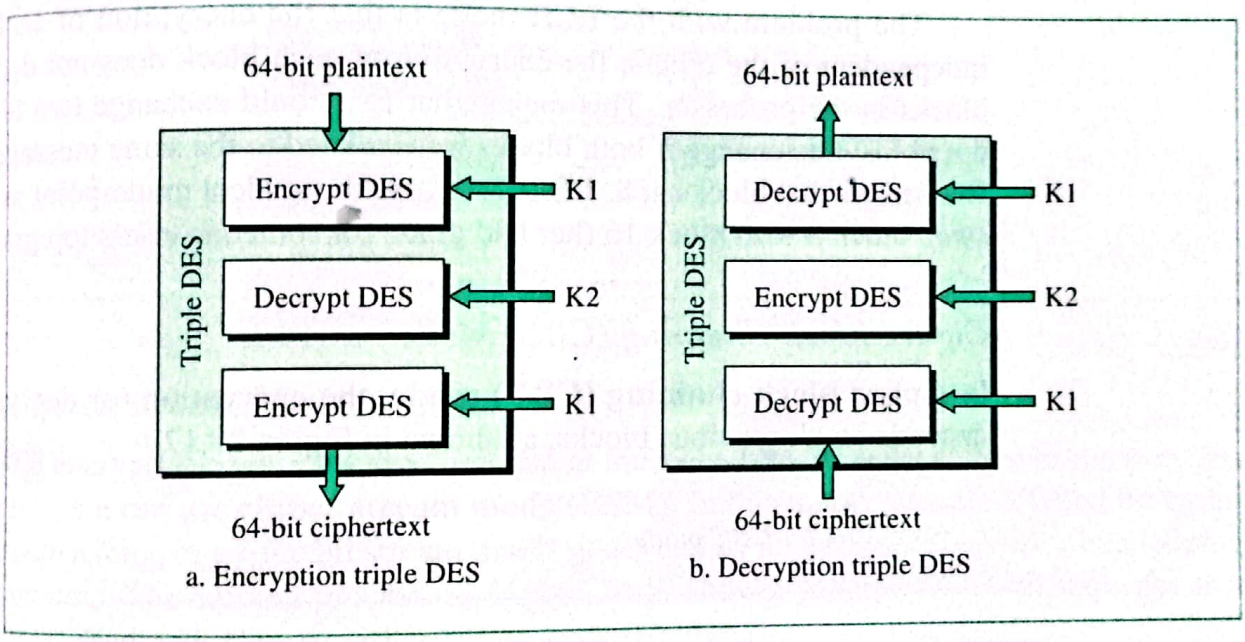


DES takes the data and chops them into 8-byte segments. However, the encryption and the key are the same for each segment. So if the data are four equal segments, the result is also four equal segments.

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.

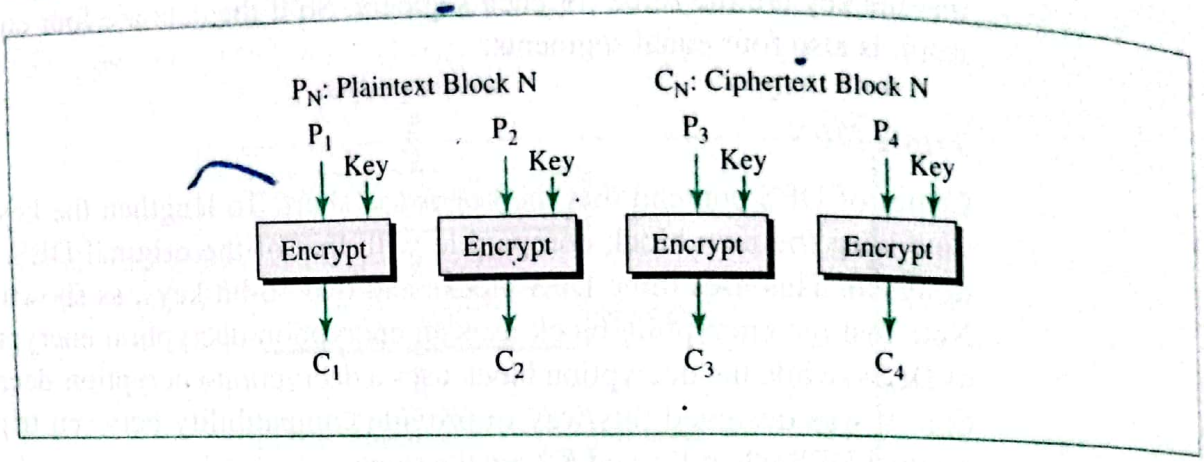
Operation Modes

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*



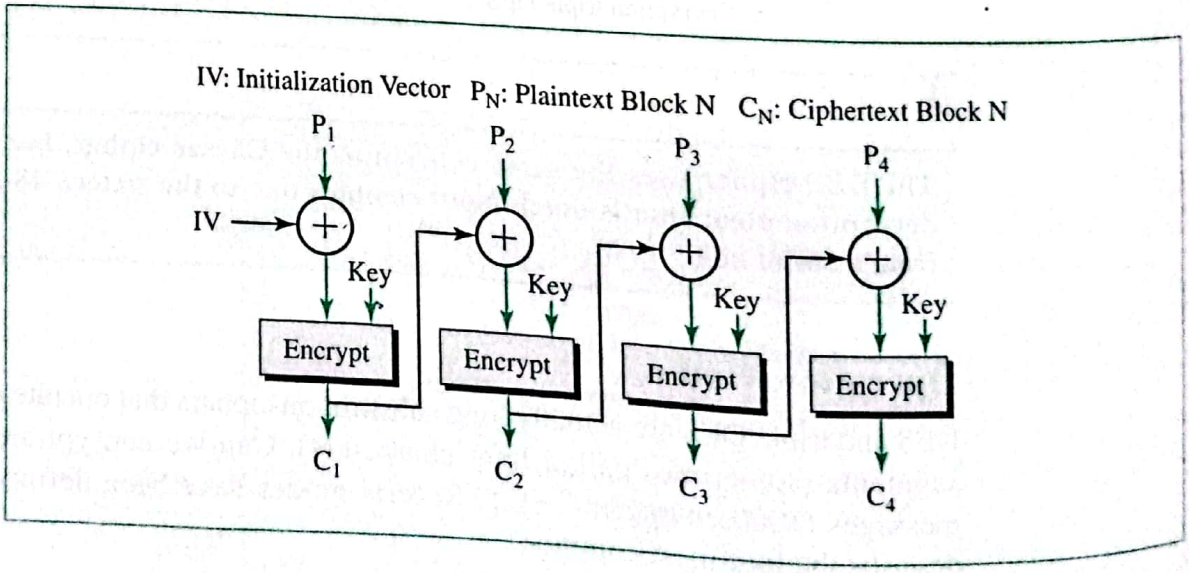
block is independent of the other blocks in ECB mode. Although the figure shows only four blocks, ECB is designed to handle many more.

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*



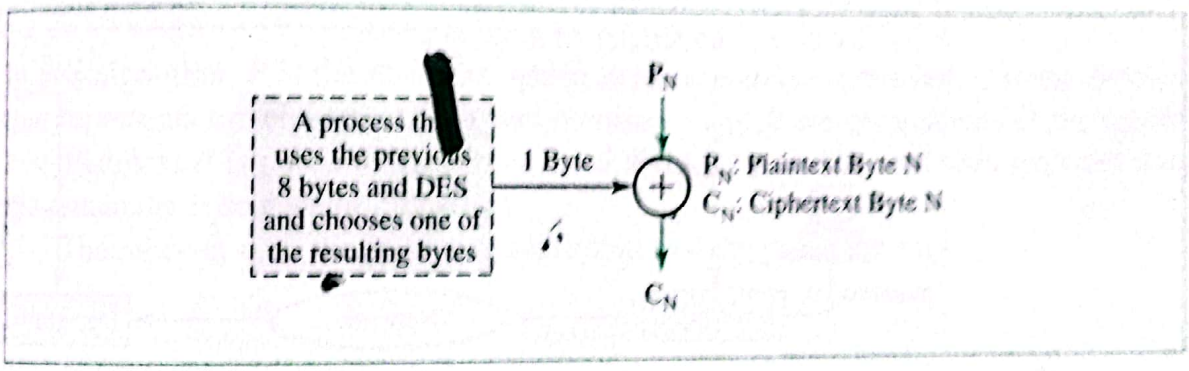
For example, to encrypt the second plaintext block (P_2), we first XOR it with the first ciphertext block (C_1) and then pass it through the encryption process. In this way, C_2 depends on C_1. If someone exchanges C_1 with C_3, for example, C_2 will

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.

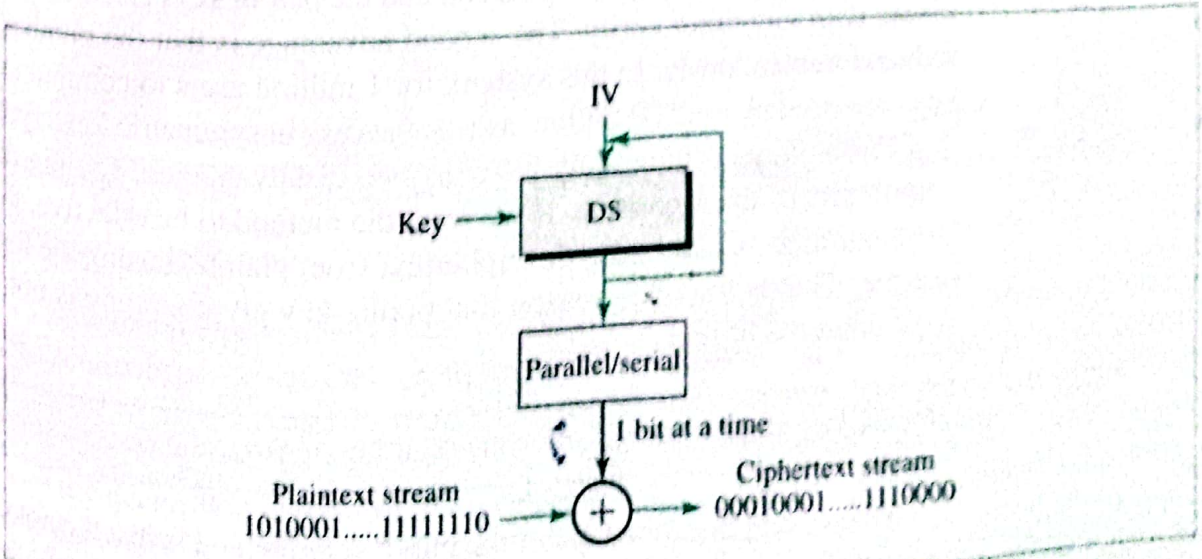
Figure 29.18 *CFM*



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*

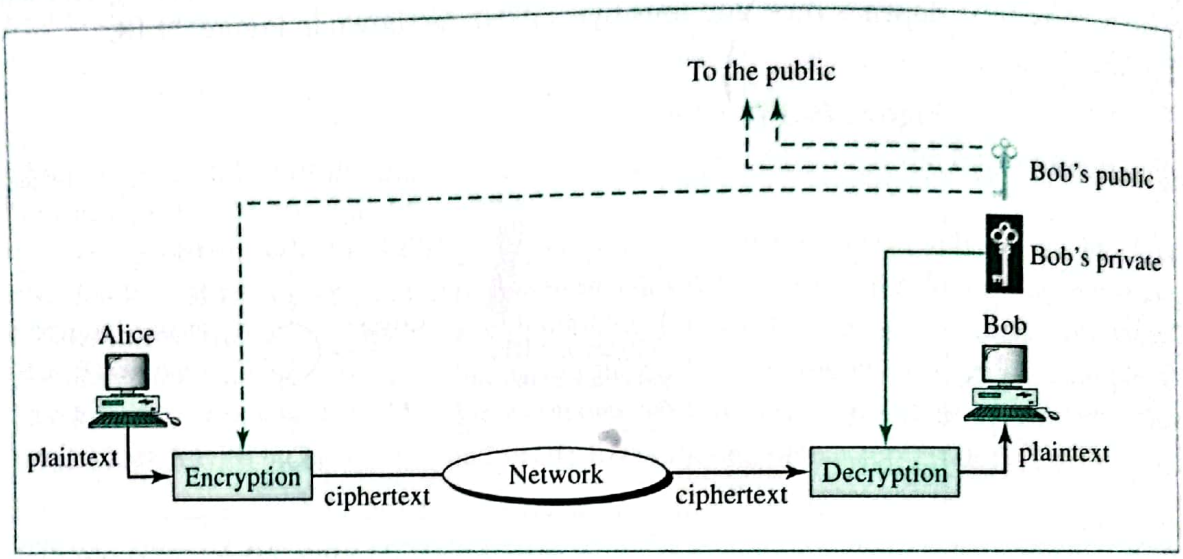


29.3 PUBLIC-KEY CRYPTOGRAPHY

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:

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

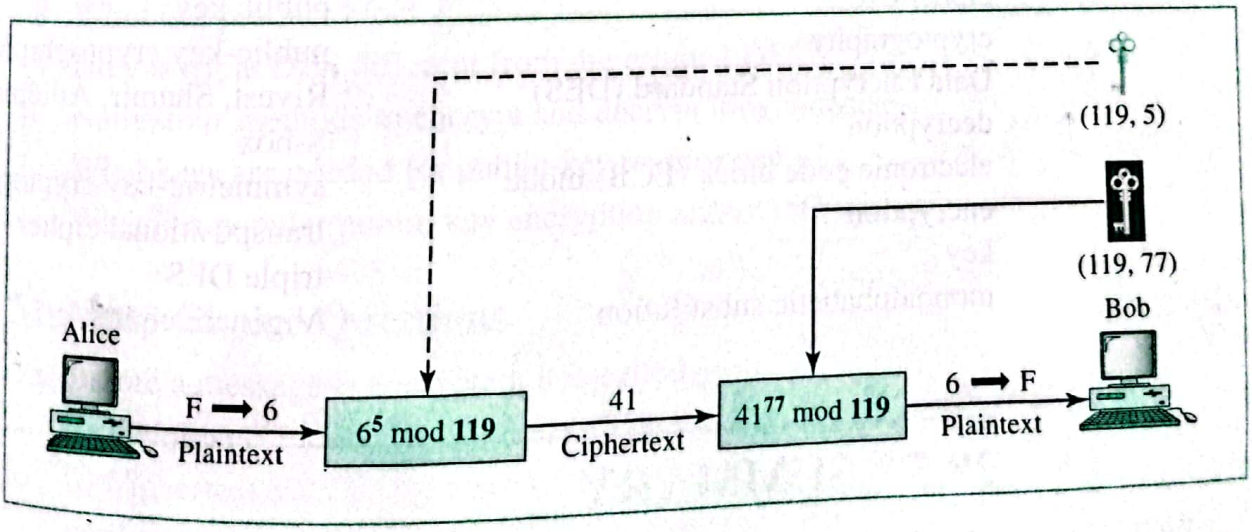
In this algorithm, P is the plaintext, which is represented as a number; C is the number that represents the ciphertext. The two numbers e and N are components of the public key. Plaintext P is raised to the power e and divided by N. The mod term indicates that the remainder is sent as the ciphertext.

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.