Now use the notation that $X[j]$ denotes the jth bit of the b-bit quantity X. Then

$$P_1[i] = \text{IV}[i] \oplus \text{D}(K, C_1)[i]$$

Then, using the properties of XOR, we can state

$$P_1[i]' = \text{IV}[i]' \oplus \text{D}(K, C_1)[i]$$

where the prime notation denotes bit complementation. This means that if an opponent can predictably change bits in IV, the corresponding bits of the received value of P_1 can be changed.

Cipher Feedback Mode

It is possible to convert any block cipher into a stream cipher by using the **cipher feedback (CFB) mode**. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted using 8 bits. If more than 8 bits are used, transmission capacity is wasted.

Figure 2.10 depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is s bits; a common value is $s = 8$. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext.

First, consider encryption. The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first unit of plaintext P_1 to produce the first unit of ciphertext C_1, which is then transmitted. In addition, the contents of the shift register are shifted left by s bits, and C_1 is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext units have been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the *encryption* function that is used, not the decryption function. This is easily explained. Let $\text{S}_s(X)$ be defined as the most significant s bits of X. Then

$$C_1 = P_1 \oplus \text{S}_s[\text{E}(K, \text{IV})]$$

Therefore,

$$P_1 = C_1 \oplus \text{S}_s[E(K, \text{IV})]$$

The same reasoning holds for subsequent steps in the process.

Counter Mode

Although interest in the **counter mode (CTR)** has increased recently, with applications to ATM (asynchronous transfer mode) network security and IPSec (IP security), this mode was proposed early on (e.g., [DIFF79]).

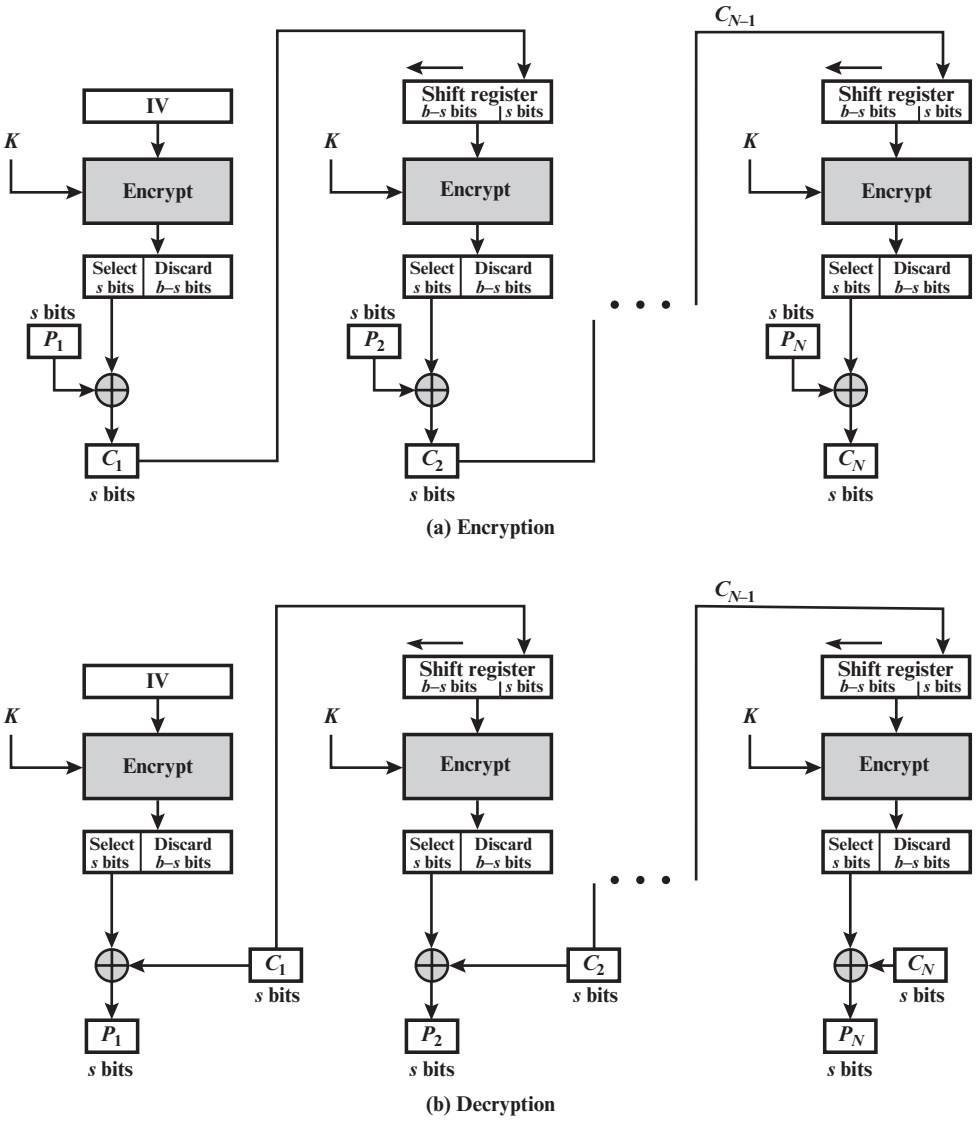

Figure 2.10 *s*-bit Cipher Feedback (CFB) Mode

Figure 2.11 depicts the CTR mode. A counter equal to the plaintext block size is used. The only requirement stated in NIST Special Publication 800-38A is that the counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^b, where b is the block size). For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block.

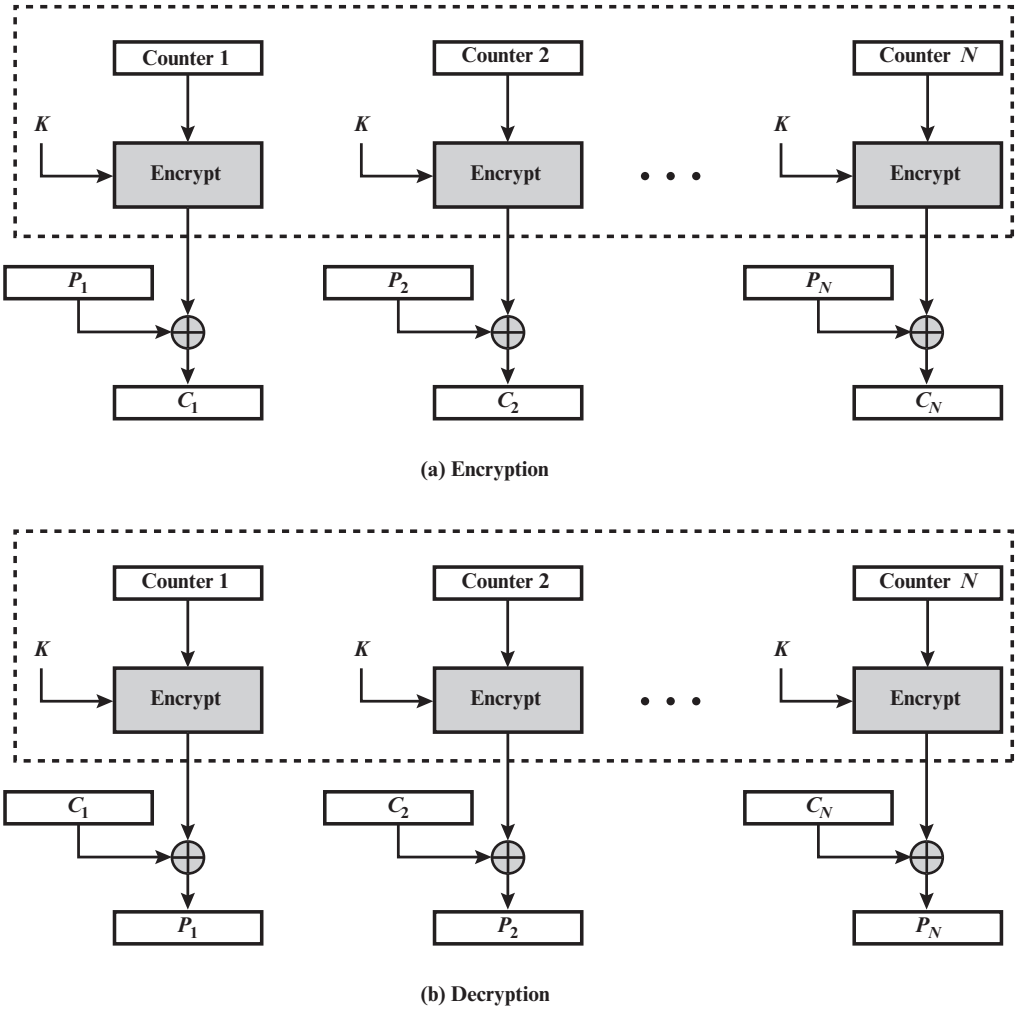

Figure 2.11 Counter (CTR) Mode

[LIPM00] lists the following advantages of CTR mode.

- **Hardware efficiency:** Unlike the chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext. For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block. This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption or decryption. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved.
- **Software efficiency:** Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features (such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions) can be effectively utilized.

- **Preprocessing:** The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions in Figure 2.11. When the plaintext or ciphertext input is presented, then the only computation is a series of XORs. Such a strategy greatly enhances throughput.
- **Random access:** The ith block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block C_i cannot be computed until the $i - 1$ prior block are computed. There may be applications in which a ciphertext is stored, and it is desired to decrypt just one block; for such applications, the random access feature is attractive.
- **Provable security:** It can be shown that CTR is at least as secure as the other modes discussed in this section.
- **Simplicity:** Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES. In addition, the decryption key scheduling need not be implemented.

2.6 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

Advanced Encryption Standard (AES)	cryptography	keystream
block cipher	Data Encryption Standard (DES)	link encryption
brute-force attack	decryption	plaintext
cipher block chaining (CBC) mode	electronic codebook (ECB) mode	session key
cipher feedback (CFB) mode	encryption	stream cipher
ciphertext	end-to-end encryption	subkey
counter mode (CTR)	Feistel cipher	symmetric encryption
cryptanalysis	key distribution	triple DES (3DES)

Review Questions

2.1 What is symmetric encryption? What are the two requirements for secure use of symmetric encryption?

2.2 What is cryptanalysis? Summarize the various types of cryptanalytic attacks on encrypted messages.

2.3 List the parameters of a symmetric block cipher for greater security.

2.4 What is a block cipher? Name the important symmetric block ciphers.

2.5 Describe the data encryption algorithm for 64-bit length plaintext and 56-bit length key.

2.6 Describe the encryption and decryption of triple DES.

2.7 What are the advantages and disadvantages of triple DES?

2.8 List the important design criteria for a stream cipher.

Problems

2.1 This problem uses a real-world example of a symmetric cipher, from an old U.S. Special Forces manual (public domain). The document, filename *SpecialForces.pdf*, is available at box.com/NetSec6e.

a. Using the two keys (memory words) *cryptographic* and *network security*, encrypt the following message:

> Be at the third pillar from the left outside the lyceum theatre tonight at seven. If you are distrustful bring two friends.

Make reasonable assumptions about how to treat redundant letters and excess letters in the memory words and how to treat spaces and punctuation. Indicate what your assumptions are. *Note:* The message is from the Sherlock Holmes novel *The Sign of Four*.

b. Decrypt the ciphertext. Show your work.

c. Comment on when it would be appropriate to use this technique and what its advantages are.

2.2 Consider a very simple symmetric block encryption algorithm in which 64-bit blocks of plaintext are encrypted using a 128-bit key. Encryption is defined as

$$C = (P \oplus K_1) \boxplus K_0$$

where C = ciphertext, K = secret key, K_0 = leftmost 64 bits of K, K_1 = rightmost 64 bits of K, $\oplus$ = bitwise exclusive OR, and $\boxplus$ is addition mod 2^{64}.

a. Show the decryption equation. That is, show the equation for P as a function of C, K_0, and K_1.

b. Suppose an adversary has access to two sets of plaintexts and their corresponding ciphertexts and wishes to determine K. We have the two equations:

$$C = (P \oplus K_1) \boxplus K_0; C' = (P' \oplus K_1) \boxplus K_0$$

First, derive an equation in one unknown (e.g., K_0). Is it possible to proceed further to solve for K_0?

2.3 Perhaps the simplest "serious" symmetric block encryption algorithm is the Tiny Encryption Algorithm (TEA). TEA operates on 64-bit blocks of plaintext using a 128-bit key. The plaintext is divided into two 32-bit blocks (L_0, R_0), and the key is divided into four 32-bit blocks (K_0, K_1, K_2, K_3). Encryption involves repeated application of a pair of rounds, defined as follows for rounds i and $i + 1$:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \boxplus \mathrm{F}(R_{i-1}, K_0, K_1, \delta_i) \\ L_{i+1} &= R_i \\ R_{i+1} &= L_i \boxplus \mathrm{F}(R_i, K_2, K_3, \delta_{i+1}) \end{aligned}$$

where F is defined as

$$\mathrm{F}(M, K_j, K_k, \delta_i) = ((\mathrm{M} << 4) \boxplus K_j) \oplus ((\mathrm{M} >> 5) \boxplus K_k) \oplus (M \boxplus \delta_i)$$

and where the logical shift of x by y bits is denoted by $x << y$, the logical right shift of x by y bits is denoted by $x >> y$, and δ_i is a sequence of predetermined constants.

a. Comment on the significance and benefit of using the sequence of constants.

b. Illustrate the operation of TEA using a block diagram or flow chart type of depiction.

c. If only one pair of rounds is used, then the ciphertext consists of the 64-bit block (L_2, R_2). For this case, express the decryption algorithm in terms of equations.
d. Repeat part (c) using an illustration similar to that used for part (b).

2.4 Is the DES decryption the inverse of DES encryption? Justify your answer.

2.5 Consider a Feistel cipher composed of 14 rounds with block length 128 bits and key length 128 bits. Suppose that, for a given k, the key scheduling algorithm determines values for the first seven round keys, $k_1, k_2, \ldots, k_8$, and then sets

$$k_8 = k_7, k_9 = k_6, k_{10} = k_5, \ldots, k_{14} = k_1$$

Suppose you have a ciphertext c. Explain how, with access to an encryption oracle, you can decrypt c and determine m using just a single oracle query. This shows that such a cipher is vulnerable to a chosen plaintext attack. (An encryption oracle can be thought of as a device that, when given a plaintext, returns the corresponding ciphertext. The internal details of the device are not known to you, and you cannot break open the device. You can only gain information from the oracle by making queries to it and observing its responses.)

2.6 For any block cipher, the fact that it is a nonlinear function is crucial to its security. To see this, suppose that we have a linear block cipher EL that encrypts 256-bit blocks of plaintext into 256-bit blocks of ciphertext. Let $\text{EL}(k, m)$ denote the encryption of a 256-bit message m under a key k (the actual bit length of k is irrelevant). Thus,

$$\text{EL}(k, [m_1 \oplus m_2]) = \text{EL}(k, m_1) \oplus \text{EL}(k, m_2) \text{ for all 256-bit patterns } m_1, m_2$$

Describe how, with 256 chosen ciphertexts, an adversary can decrypt any ciphertext without knowledge of the secret key k. (A "chosen ciphertext" means that an adversary has the ability to choose a ciphertext and then obtain its decryption. Here, you have 256 plaintext–ciphertext pairs to work with, and you have the ability to choose the value of the ciphertexts.)

2.7 Suppose you have a true random bit generator where each bit in the generated stream has the same probability of being a 0 or 1 as any other bit in the stream and that the bits are not correlated; that is, the bits are generated from identical independent distribution. However, the bit stream is biased. The probability of a 1 is $0.5 - \delta$ and the probability of a 0 is $0.5 + \delta$ where $0 < \delta < 0.5$. A simple deskewing algorithm is as follows: Examine the bit stream as a sequence of nonoverlapping pairs. Discard all 00 and 11 pairs. Replace each 01 pair with 0 and each 10 pair with 1.
a. What is the probability of occurrence of each pair in the original sequence?
b. What is the probability of occurrence of 0 and 1 in the modified sequence?
c. What is the expected number of input bits to produce x output bits?
d. Suppose that the algorithm uses overlapping successive bit pairs instead of nonoverlapping successive bit pairs. That is, the first output bit is based on input bits 1 and 2, the second output bit is based on input bits 2 and 3, and so on. What can you say about the output bit stream?

2.8 Another approach to deskewing is to consider the bit stream as a sequence of nonoverlapping groups of n bits each and output the parity of each group. That is, if a group contains an odd number of ones, the output is 1; otherwise the output is 0.
a. Express this operation in terms of a basic Boolean function.
b. Assume, as in the Problem 2.7, that the probability of a 1 is $0.5 - \delta$. If each group consists of 2 bits, what is the probability of an output of 1?
c. If each group consists of 3 bits, what is the probability of an output of 1?
d. Generalize the result to find the probability of an output of 1 for input groups of n bits.

2.9 Is it appropriate to reuse keys in RC4? Why or why not?

2.10 RC4 has a secret internal state which is a permutation of all the possible values of the vector **S** and the two indices *i* and *j*.

- **a.** Using a straightforward scheme to store the internal state, how many bits are used?
- **b.** Suppose we think of it from the point of view of how much information is represented by the state. In that case, we need to determine how many different states there are, then take the log to the base 2 to find out how many bits of information this represents. Using this approach, how many bits would be needed to represent the state?

2.11 Alice and Bob agree to communicate privately via e-mail using a scheme based on RC4, but they want to avoid using a new secret key for each transmission. Alice and Bob privately agree on a 128-bit key k. To encrypt a message m consisting of a string of bits, the following procedure is used.

1. Choose a random 80-bit value v
2. Generate the ciphertext $c = \text{RC4}(v \| k) \oplus m$
3. Send the bit string $(v \| c)$
   - **a.** Suppose Alice uses this procedure to send a message m to Bob. Describe how Bob can recover the message m from $(v \| c)$ using k.
   - **b.** If an adversary observes several values $(v_1 \| c_1), (v_2 \| c_2), \ldots$ transmitted between Alice and Bob, how can he or she determine when the same key stream has been used to encrypt two messages?

2.12 With the ECB mode, if there is an error in a block of the transmitted ciphertext, only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted C_1 (Figure 2.9) obviously corrupts P_1 and P_2.

- **a.** Are any blocks beyond P_2 affected?
- **b.** Suppose that there is a bit error in the source version of P_1. Through how many ciphertext blocks is this error propagated? What is the effect at the receiver?

2.13 Is it possible to perform decryption operations in parallel on multiple blocks of ciphertext in CBC mode? How about encryption?

2.14 Why should the IV in CBC be protected?

2.15 CBC-Pad is a block cipher mode of operation used in the RC5 block cipher, but it could be used in any block cipher. CBC-Pad handles plaintext of any length. The ciphertext is longer than the plaintext by at most the size of a single block. Padding is used to assure that the plaintext input is a multiple of the block length. It is assumed that the original plaintext is an integer number of bytes. This plaintext is padded at the end by from 1 to *bb* bytes, where *bb* equals the block size in bytes. The pad bytes are all the same and set to a byte that represents the number of bytes of padding. For example, if there are 8 bytes of padding, each byte has the bit pattern `00001000`. Why not allow zero bytes of padding? That is, if the original plaintext is an integer multiple of the block size, why not refrain from padding?

2.16 Padding may not always be appropriate. For example, one might wish to store the encrypted data in the same memory buffer that originally contained the plaintext. In that case, the ciphertext must be the same length as the original plaintext. A mode for that purpose is the ciphertext stealing (CTS) mode. Figure 2.12a shows an implementation of this mode.

- **a.** Explain how it works.
- **b.** Describe how to decrypt C_{n-1} and C_n.

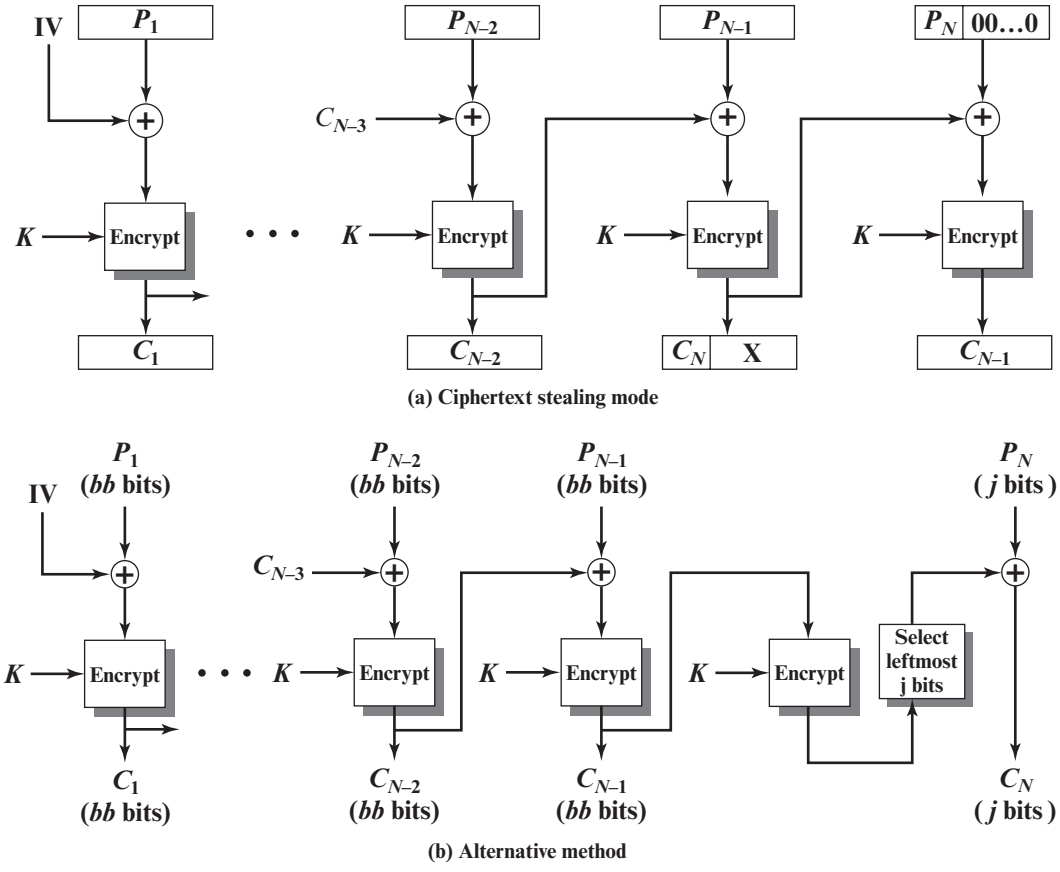

Figure 2.12 Block Cipher Modes for Plaintext not a Multiple of Block Size

2.17 Figure 2.12b shows an alternative to CTS for producing ciphertext of equal length to the plaintext when the plaintext is not an integer multiple of the block size.
- **a.** Explain the algorithm.
- **b.** Explain why CTS is preferable to this approach illustrated in Figure 2.12b.

2.18 If a bit error occurs in the transmission of a ciphertext character in 8-bit CFB mode, how far does the error propagate?

CHAPTER 3

Public-Key Cryptography and Message Authentication

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Define the term *message authentication code*.
- List and explain the requirements for a message authentication code.
- Explain why a hash function used for message authentication needs to be secured.
- Understand the differences among preimage resistant, second preimage resistant, and collision resistant properties.
- Understand the operation of SHA-512.
- Present an overview of HMAC.
- Present an overview of the basic principles of public-key cryptosystems.
- Explain the two distinct uses of public-key cryptosystems.
- Present an overview of the RSA algorithm.
- Define Diffie–Hellman key exchange.
- Understand the man-in-the-middle attack.

In addition to message confidentiality, message authentication is an important network security function. This chapter examines three aspects of message authentication. First, we look at the use of message authentication codes and hash functions to provide message authentication. Then we look at public-key encryption principles and two specific public-key algorithms. These algorithms are useful in the exchange of conventional encryption keys. Then we look at the use of public-key encryption to produce digital signatures, which provides an enhanced form of message authentication.

3.1 APPROACHES TO MESSAGE AUTHENTICATION

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

A message, file, document, or other collection of data is said to be authentic when it is genuine and comes from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received messages are authentic.[1] The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties. All of these concerns come under the category of data integrity as described in Chapter 1.

[1]For simplicity, for the remainder of this chapter, we refer to *message authentication.* By this we mean both authentication of transmitted messages and of stored data (*data authentication*).