

6.2 ELECTRONIC CODE BOOK

A block cipher takes a fixed-length block of text of length b bits and a key as input and produces a b -bit block of ciphertext. If the amount of plaintext to be encrypted is greater than b bits, then the block cipher can still be used by breaking the plaintext

up into b -bit blocks. When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise. To apply a block cipher in a variety of applications, five *modes of operation* have been defined by NIST (SP 800-38A). In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. The five modes are intended to cover a wide variety of applications of encryption for which a block cipher could be used. These modes are intended for use with any symmetric block cipher, including triple DES and AES. The modes are summarized in Table 6.1 and described in this and the following sections.

The simplest mode is the **electronic codebook (ECB)** mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key (Figure 6.3). The term *codebook* is used because, for a given key, there is a unique ciphertext for every b -bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible b -bit plaintext pattern showing its corresponding ciphertext.

For a message longer than b bits, the procedure is simply to break the message into b -bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key. In Figure 6.3, the plaintext (padded as necessary) consists of a sequence of b -bit blocks, P_1, P_2, \dots, P_N ; the

Table 6.1 Block Cipher Modes of Operation

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	<ul style="list-style-type: none"> Secure transmission of single values (e.g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	<ul style="list-style-type: none"> General-purpose block-oriented transmission Authentication
Cipher Feedback (CFB)	Input is processed s bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	<ul style="list-style-type: none"> General-purpose stream-oriented transmission Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	<ul style="list-style-type: none"> Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	<ul style="list-style-type: none"> General-purpose block-oriented transmission Useful for high-speed requirements

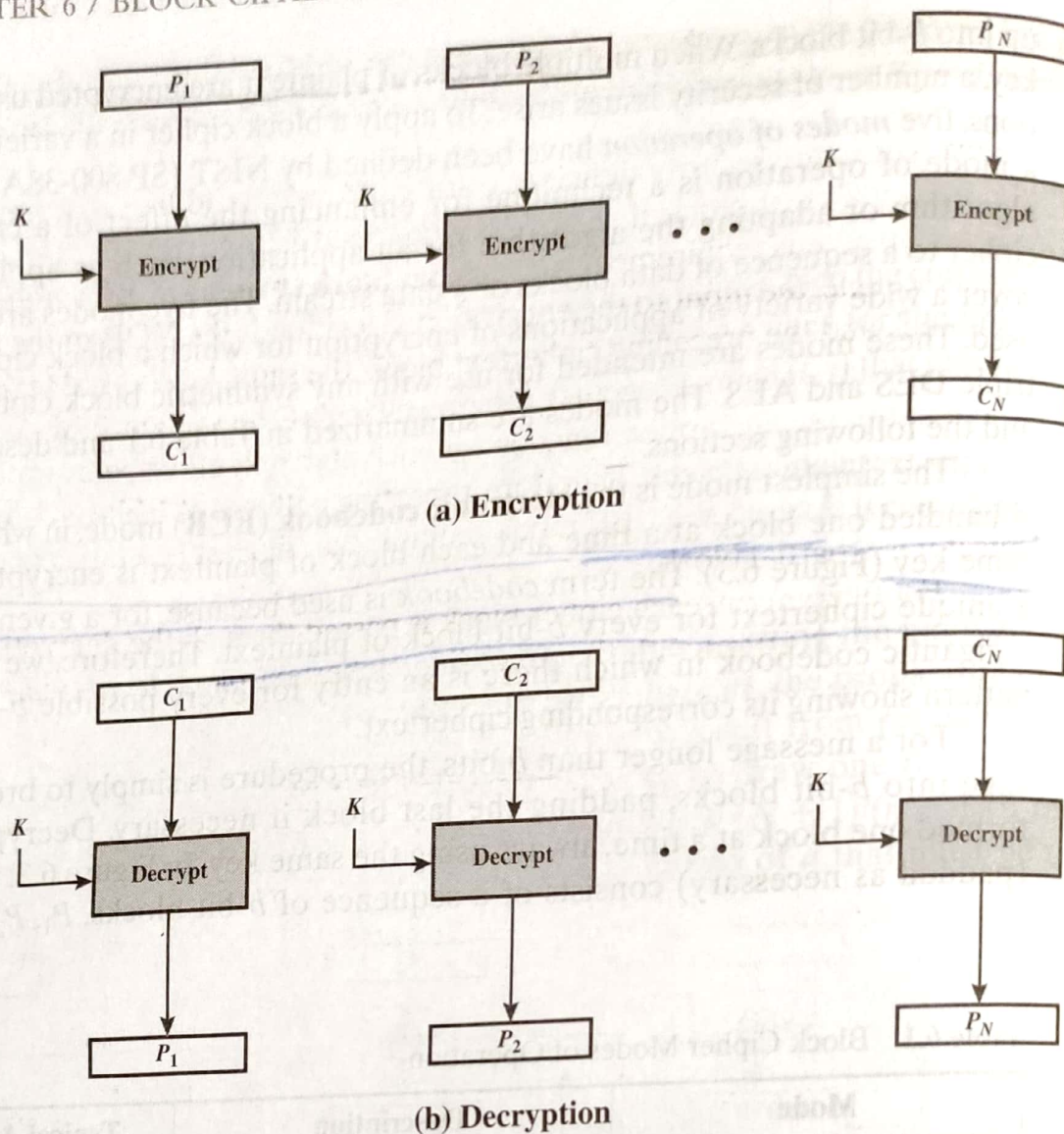


Figure 6.3 Electronic Codebook (ECB) Mode

corresponding sequence of ciphertext blocks is C_1, C_2, \dots, C_N . We can define ECB mode as follows.

ECB	$C_j = E(K, P_j)$	$j = 1, \dots, N$	$P_j = D(K, C_j)$	$j = 1, \dots, N$
-----	-------------------	-------------------	-------------------	-------------------

The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES or AES key securely, ECB is the appropriate mode to use.

The most significant characteristic of ECB is that if the same b -bit block of plaintext appears more than once in the message, it always produces the same ciphertext.

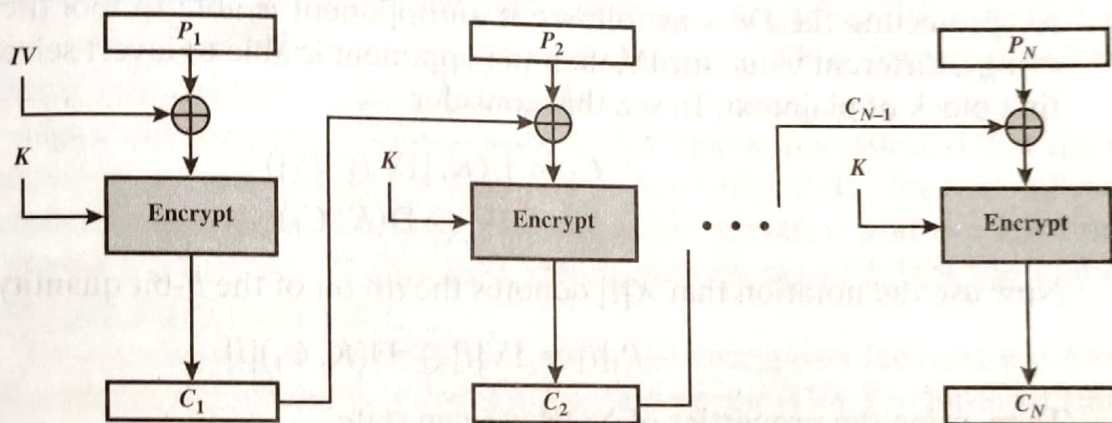
For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with. If the message has repetitive elements with a period of repetition a multiple of b bits, then these elements can be identified by the analyst. This may help in the analysis or may provide an opportunity for substituting or rearranging blocks.

6.3 CIPHER BLOCK CHAINING MODE

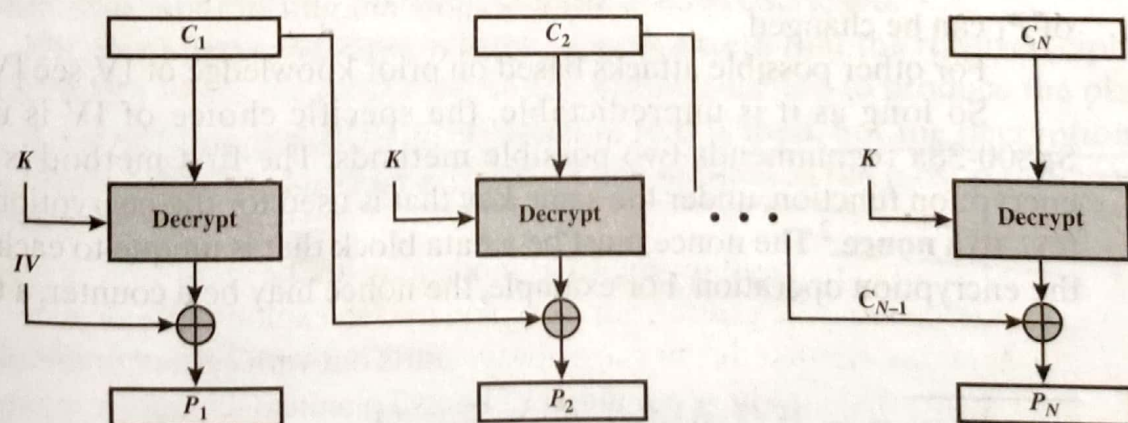
To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the **cipher block chaining (CBC) mode** (Figure 6.4). In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of b bits are not exposed. As with the ECB mode, the CBC mode requires that the last block be padded to a full b bits if it is a partial block.

For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block. To see that this works, we can write

$$C_j = E(K, [C_{j-1} \oplus P_j])$$



(a) Encryption



(b) Decryption

Figure 6.4 Cipher Block Chaining (CBC) Mode

Then

$$D(K, C_j) = D(K, E(K, [C_{j-1} \oplus P_j]))$$

$$D(K, C_j) = C_{j-1} \oplus P_j$$

$$C_{j-1} \oplus D(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext. The IV is a data block that is that same size as the cipher block. We can define CBC mode as

CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \dots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \dots, N$
-----	--	--

The IV must be known to both the sender and receiver but be unpredictable by a third party. In particular, for any given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV. For maximum security, the IV should be protected against unauthorized changes. This could be done by sending the IV using ECB encryption. One reason for protecting the IV is as follows: If an opponent is able to fool the receiver into using a different value for IV, then the opponent is able to invert selected bits in the first block of plaintext. To see this, consider

$$C_1 = E(K, [IV \oplus P_1])$$

$$P_1 = IV \oplus D(K, C_1)$$

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

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

Then, using the properties of XOR, we can state

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

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.

For other possible attacks based on prior knowledge of IV, see [VOYD83].

So long as it is unpredictable, the specific choice of IV is unimportant. Sp800-38a recommends two possible methods: The first method is to apply the encryption function, under the same key that is used for the encryption of the plaintext, to a **nonce**.² The nonce must be a data block that is unique to each execution of the encryption operation. For example, the nonce may be a counter, a timestamp, or

a message number. The second method is to generate a random data block using a random number generator.

In conclusion, because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than b bits.

In addition to its use to achieve confidentiality, the CBC mode can be used for authentication. This use is described in Chapter 12.

6.4 CIPHER FEEDBACK MODE

For AES, DES, or any block cipher, encryption is performed on a block of b bits. In the case of DES, $b = 64$ and in the case of AES, $b = 128$. However, it is possible to convert a block cipher into a stream cipher, using one of the three modes to be discussed in this and the next two sections: **cipher feedback** (CFB) mode, **output feedback** (OFB) mode, and **counter** (CTR) 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 to produce a ciphertext output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.

Figure 6.5 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. In this case, rather than blocks of b bits, the plaintext is divided into segments of s bits.

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 segment 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 $MSB_s(X)$ be defined as the most significant s bits of X . Then

$$C_1 = P_1 \oplus MSB_s[E(K, IV)]$$

Therefore, by rearranging terms:

$$P_1 = C_1 \oplus MSB_s[E(K, IV)]$$

The same reasoning holds for subsequent steps in the process.

We can define CFB mode as follows.

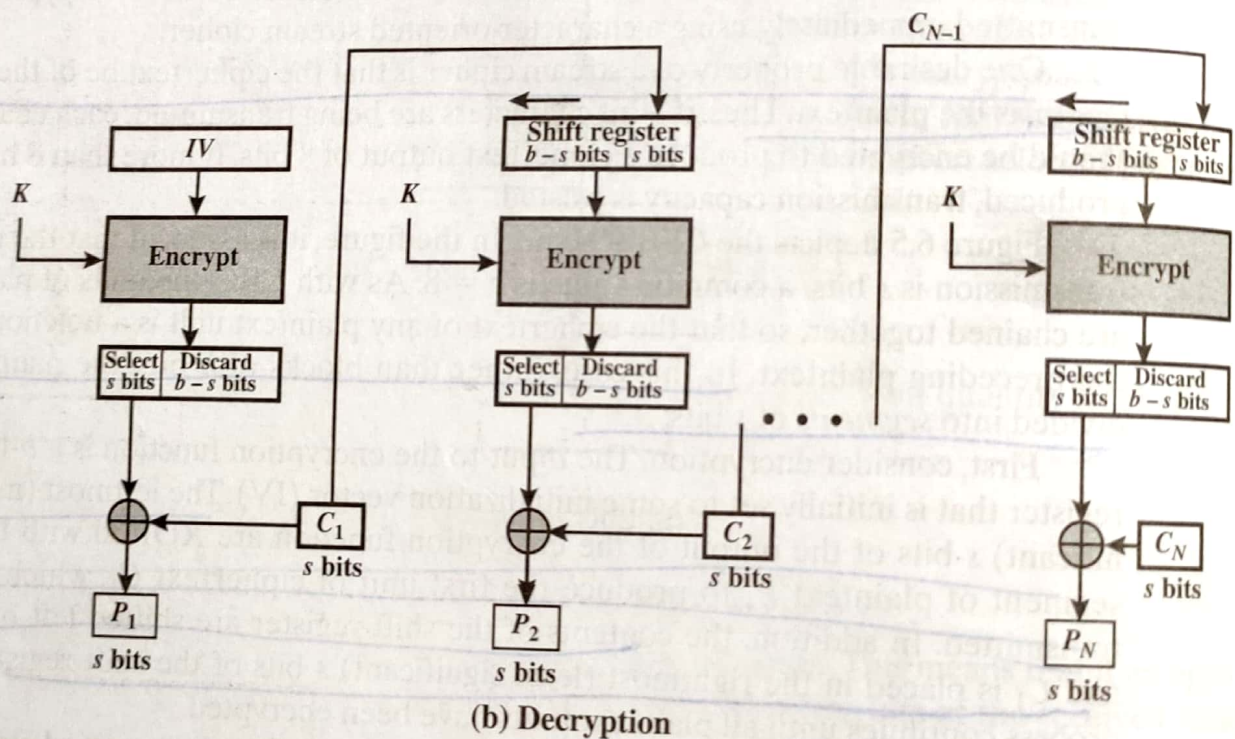
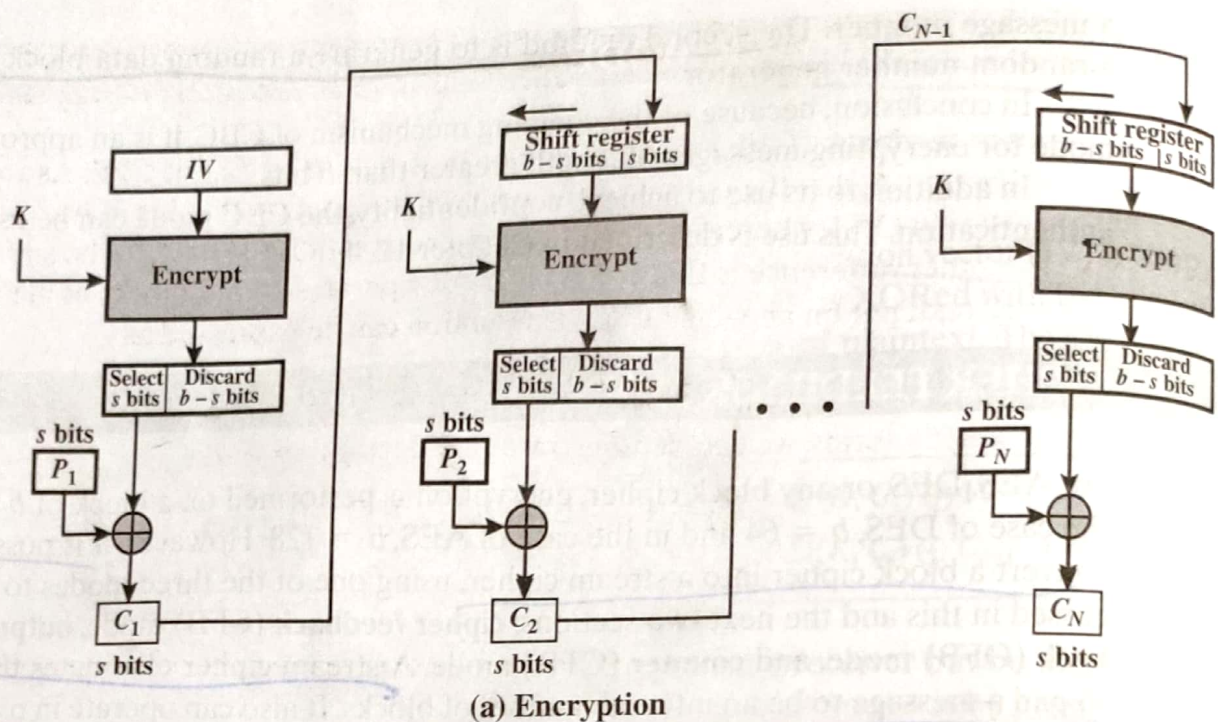


Figure 6.5 s-bit Cipher Feedback (CFB) Mode

CFB	$I_1 = IV$	$I_1 = IV$
	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$
	$O_j = E(K, I_j) \quad j = 1, \dots, N$	$O_j = E(K, I_j) \quad j = 1, \dots, N$
	$C_j = P_j \oplus \text{MSB}_s(O_j) \quad j = 1, \dots, N$	$P_j = C_j \oplus \text{MSB}_s(O_j) \quad j = 1, \dots, N$

Although CFB can be viewed as a stream cipher, it does not conform to the typical construction of a stream cipher. In a typical stream cipher, the cipher takes as input some initial value and a key and generates a stream of bits, which is then XORed with the plaintext bits (see Figure 3.1). In the case of CFB, the stream of bits that is XORed with the plaintext also depends on the plaintext.

6.5 OUTPUT FEEDBACK MODE

The **output feedback** (OFB) mode is similar in structure to that of CFB. As can be seen in Figure 6.6, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB, the ciphertext unit is fed back to the shift register. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on an s -bit subset. Encryption can be expressed as

$$C_j = P_j \oplus E(K, [C_{j-1} \oplus P_{j-1}])$$

By rearranging terms, we can demonstrate that decryption works.

$$P_j = C_j \oplus E(K, [C_{j-1} \oplus P_{j-1}])$$

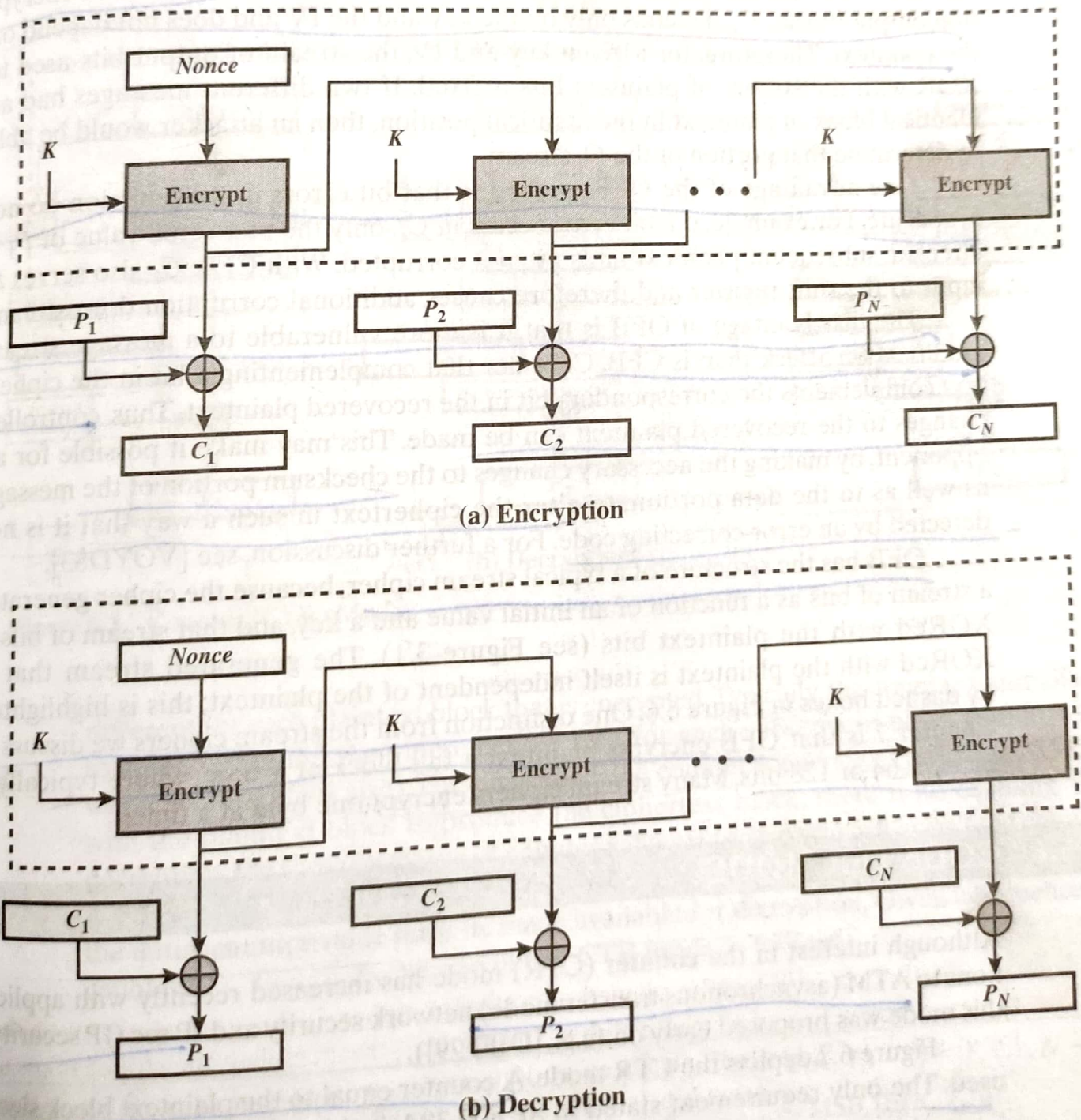


Figure 6.6 Output Feedback (OFB) Mode

We can define OFB mode as follows.

OFB	$I_1 = \text{Nonce}$	$I_1 = \text{Nonce}$
	$I_j = O_{j-1} \quad j = 2, \dots, N$	$I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \dots, N$
	$O_j = E(K, I_j) \quad j = 1, \dots, N$	$O_j = E(K, I_j) \quad j = 1, \dots, N$
	$C_j = P_j \oplus O_j \quad j = 1, \dots, N-1$	$P_j = C_j \oplus O_j \quad j = 1, \dots, N-1$
	$C_N^* = P_N^* \oplus \text{MSB}_u(O_N)$	$P_N^* = C_N^* \oplus \text{MSB}_u(O_N)$

Let the size of a block be b . If the last block of plaintext contains u bits (indicated by *), with $u < b$, the most significant u bits of the last output block O_N are used for the XOR operation; the remaining $b-u$ bits of the last output block are discarded. In the

case of OFB, the IV must be a nonce; that is, the IV must be unique to each execution of the encryption operation. The reason for this is that the sequence of encryption output blocks, O_i , depends only on the key and the IV and does not depend on the plaintext. Therefore, for a given key and IV, the stream of output bits used to XOR with the stream of plaintext bits is fixed. If two different messages had an identical block of plaintext in the identical position, then an attacker would be able to determine that portion of the O_i stream.

One advantage of the OFB method is that bit errors in transmission do not propagate. For example, if a bit error occurs in C_1 , only the recovered value of P_1 is affected; subsequent plaintext units are not corrupted. With CFB, C_1 also serves as input to the shift register and therefore causes additional corruption downstream.

The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB. Consider that complementing a bit in the ciphertext complements the corresponding bit in the recovered plaintext. Thus, controlled changes to the recovered plaintext can be made. This may make it possible for an opponent, by making the necessary changes to the checksum portion of the message as well as to the data portion, to alter the ciphertext in such a way that it is not detected by an error-correcting code. For a further discussion, see [VOYD83].

OFB has the structure of a typical stream cipher, because the cipher generates a stream of bits as a function of an initial value and a key, and that stream of bits is XORed with the plaintext bits (see Figure 3.1). The generated stream that is XORed with the plaintext is itself independent of the plaintext; this is highlighted by dashed boxes in Figure 6.6. One distinction from the stream ciphers we discuss in Chapter 7 is that OFB encrypts plaintext a full block at a time, where typically a block is 64 or 128 bits. Many stream ciphers encrypt one byte at a time.