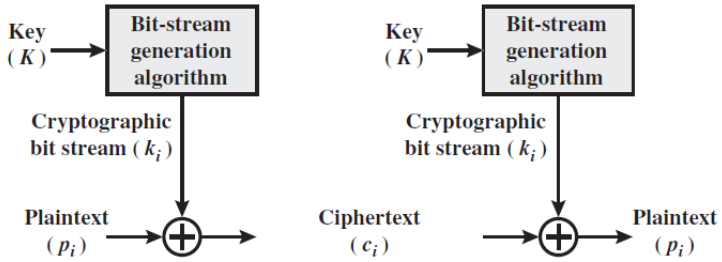
Chapter 2- Block Ciphers and data encryption standard (DES)

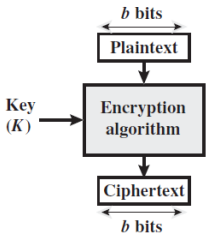
Stream Ciphers and Block Ciphers

- A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time.
    - Example of classical stream ciphers is the Vernam cipher.
- To generate the long key we need a bit-stream generator, so that the cryptographic bit stream can be produced by both users.



(a) Stream cipher using algorithmic bit-stream generator

- A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key.



(b) Block cipher

- To encrypt the blocks of the plaintext we use **modes of operation**
- Block cipher is applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers.

The Feistel Cipher

- A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits.
- There are 2^n possible different plaintext blocks and, for the encryption to be reversible, each must produce a unique ciphertext block.

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

- Feistel use the concept of a **product cipher**, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.
- In particular, Feistel proposed the use of a cipher that alternates **substitutions** and **permutations**.
- In fact, Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions
- **Diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible in order to thwart attempts to deduce the key
- **Confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.

FEISTEL CIPHER STRUCTURE

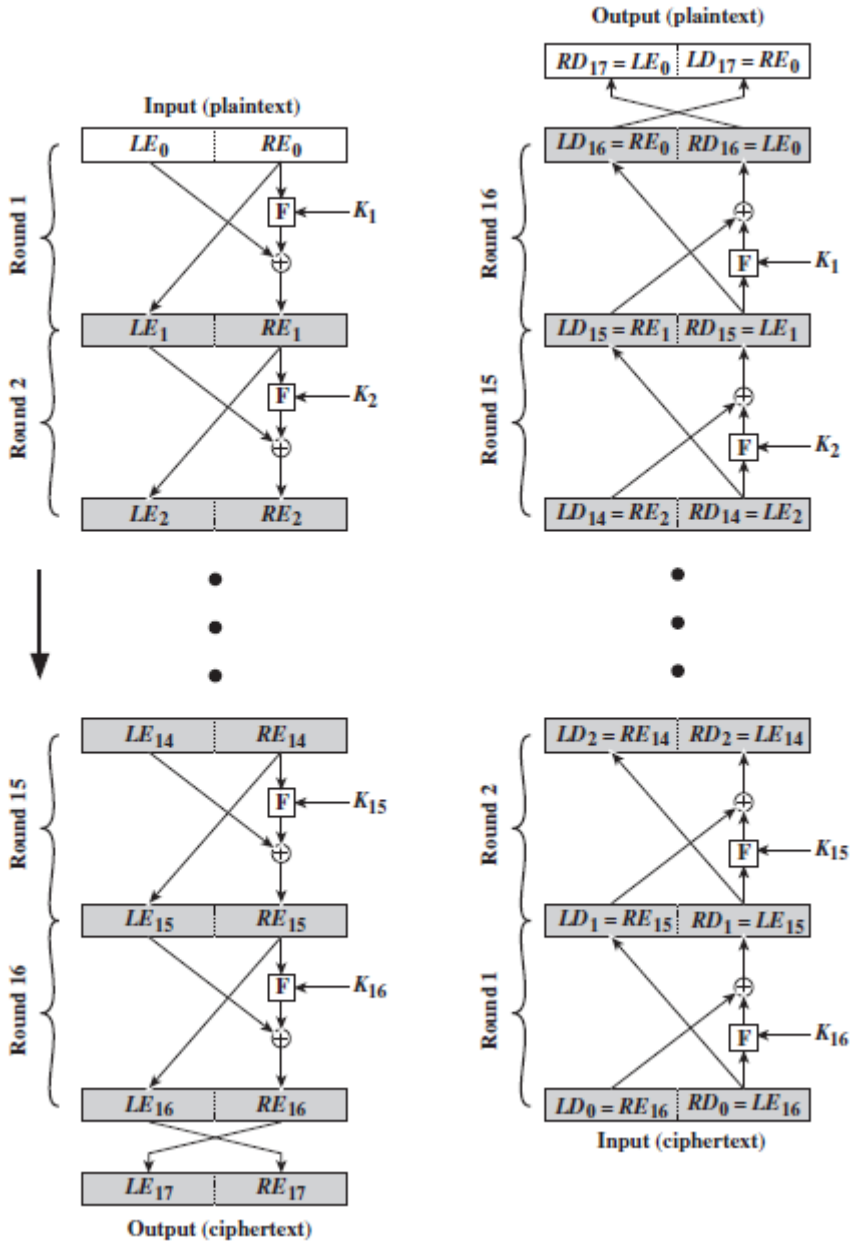
- The inputs to the encryption algorithm are a plaintext block of *2w* length bits and a key.
- The plaintext block is divided into two halves L_0, and R_0.
- The two halves of the data pass through *rounds* of processing and then combined to produce the ciphertext block.
- Each round has as inputs Li-1 and Ri-1 and derived from the previous round, as well as *ki* a subkey derived from the overall K.
- A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the *exclusive*-OR of the output of that function and the left half of the data.
- **Permutation** is performed that consists of the interchange of the two halves of the data

- The exact realization of a Feistel network depends on the choice of the following parameters and design features:

1- **Block size:** Larger block sizes mean greater security but reduced encryption/decryption speed for a given algorithm. AES uses a 128-bit block size.
2- **Key size:** Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion.
3- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
4- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
5- **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.

There are two other considerations in the design of a Feistel cipher:

6- **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
7- **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze to develop a higher level of assurance as to its strength.



THE DATA ENCRYPTION STANDARD (DES)

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST)

- The algorithm itself is referred to as the Data Encryption Algorithm (DEA).
- For DES, data are encrypted in 64-bit blocks using a 56-bit key (a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored).
- The algorithm transforms 64-bit input in a series of steps into a 64-bit output.
- The same steps, with the same key, are used to reverse the encryption.

DES has the exact structure of a Feistel cipher

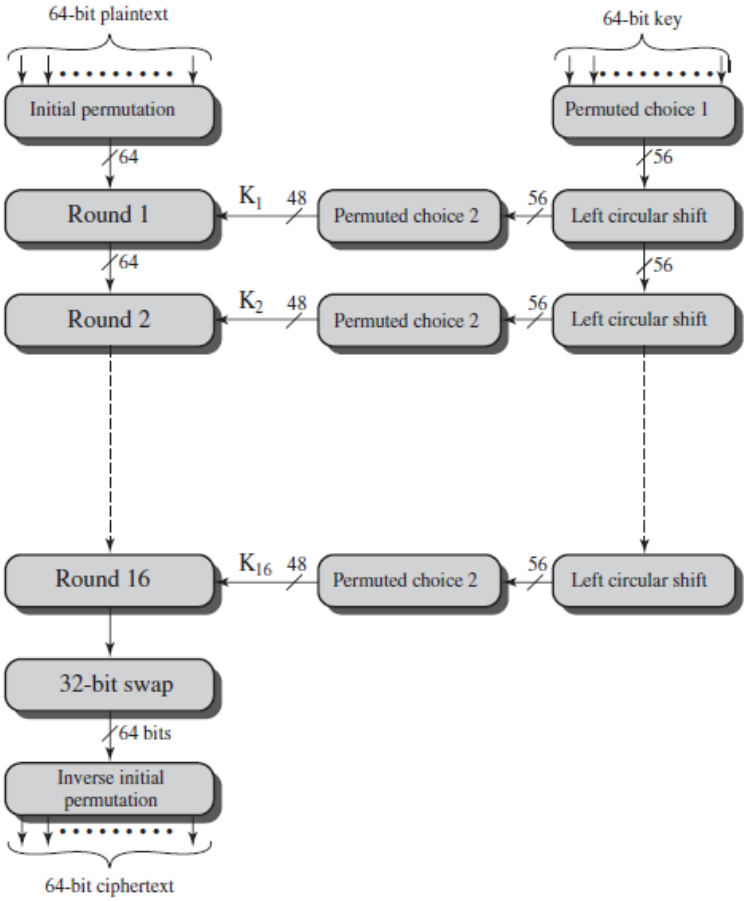


Figure 3.5 General Depiction of DES Encryption Algorithm

- Each round is illustrated as follows:

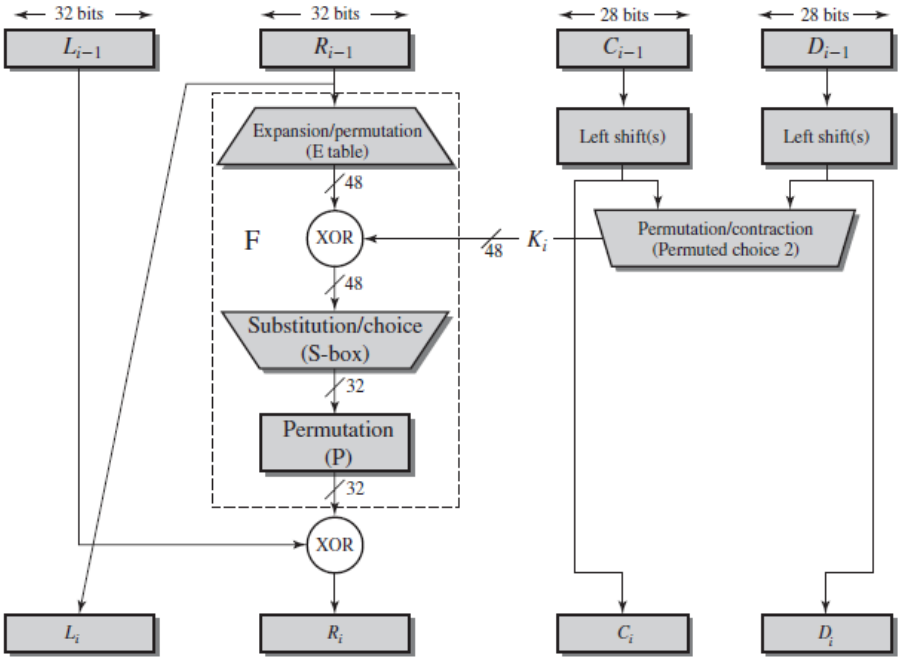


Figure 3.6 Single Round of DES Algorithm

- The round function f can be summarized as follows:

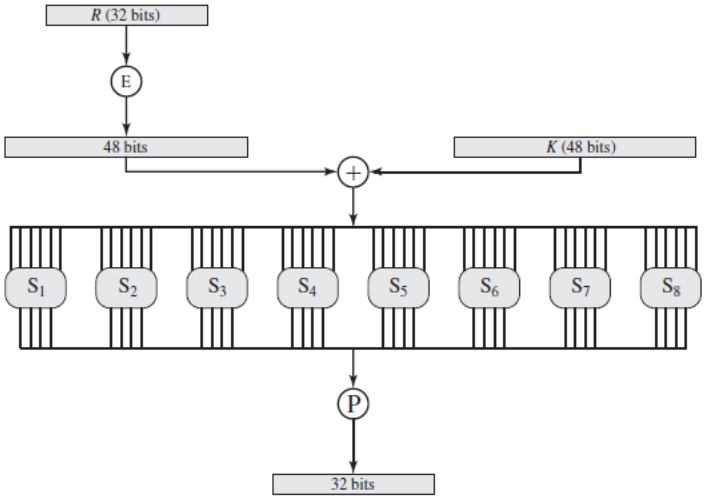


Figure 3.7 Calculation of F(R, K)

- DES uses 8 *different* s-box tables, each one receives 6-bits and outputs 4-bits.
- The initial permutation and its inverse are defined by tables, as shown below:

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

- Expansion and permutation tables are:

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

- *DES key schedule bit selections (permuted choice 1* and *permuted choice 2).*

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
above for C_i; below for D_i						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

The Avalanche Effect

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- The following table shows the result when the fourth bit of the plaintext is changed.

Table 3.6 Avalanche Effect in DES: Change in Plaintext

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Modes of operation

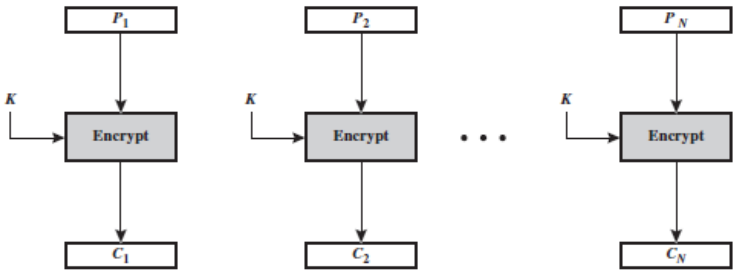
- A block cipher encrypts plaintext in fixed-size n-bit blocks (often n = 64). For messages exceeding n bits, the simplest approach is to partition the message into n-bit blocks and encrypt each separately.

Electronic codebook (ECB) mode

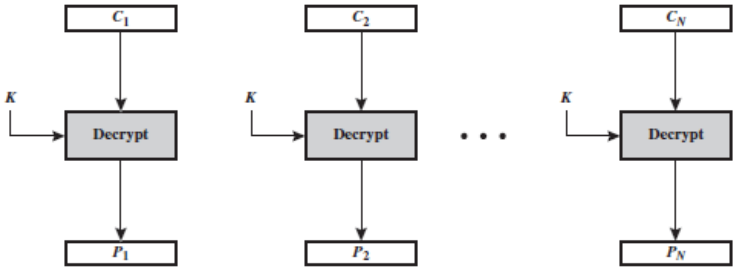
Algorithm ECB mode of operation

INPUT: k-bit key K; n-bit plaintext blocks $x_1, \ldots, x_t$.
SUMMARY: produce ciphertext blocks $c_1, \ldots, c_t$; decrypt to recover plaintext.

1. Encryption: for $1 \leq j \leq t$, $c_j \leftarrow E_K(x_j)$.
2. Decryption: for $1 \leq j \leq t$, $x_j \leftarrow E_K^{-1}(c_j)$.



(a) Encryption



(b) Decryption

Figure 6.3 Electronic Codebook (ECB) Mode

Properties of the ECB mode of operation:

1- **Identical plaintext blocks** (under the same key) result in identical ciphertext.
2- **Chaining dependencies**: blocks are enciphered independently of other blocks. Reordering ciphertext blocks results in correspondingly re-ordered plaintext blocks.
3- **Error propagation**: one or more bit errors in a single ciphertext block affect decipherment of that block only.

Note: Since ciphertext blocks are independent, malicious substitution of ECB blocks (e.g., insertion of a frequently occurring block) does not affect the decryption of adjacent blocks. Furthermore, block ciphers do not hide data patterns. For this reason, the ECB mode is not recommended for messages longer than one block.

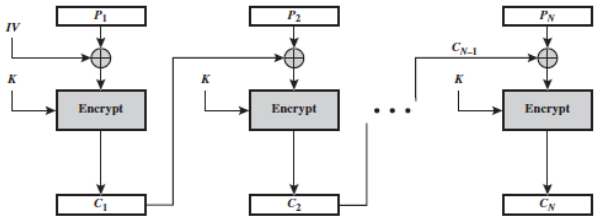
Cipher-block chaining (CBC) mode

- To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks.
- This mode involves use of an **secret** n-bit initialization vector, denoted IV.
- The IV must be known to both the sender and receiver but be unpredictable by a third party.

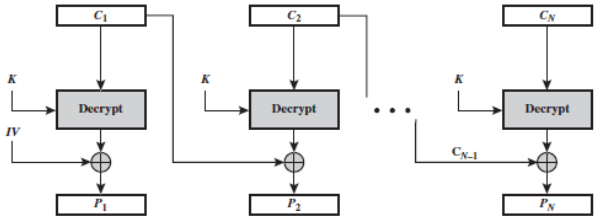
Algorithm CBC mode of operation

INPUT: k-bit key K; n-bit IV; n-bit plaintext blocks $x_1, \ldots, x_t$.
SUMMARY: produce ciphertext blocks $c_1, \ldots, c_t$; decrypt to recover plaintext.

1. Encryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$.
2. Decryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.



(a) Encryption



(b) Decryption

Properties of the CBC mode of operation:

1- **Identical plaintexts**: identical ciphertext blocks result when the same plaintext is enciphered under the same key and IV . Changing the IV , key, or first plaintext block results in different ciphertext.
2- **Chaining dependencies**: the chaining mechanism causes ciphertext cj to depend on xj and all preceding plaintext blocks. Consequently, rearranging the order of ciphertext blocks affects decryption. Proper decryption of a correct ciphertext block requires a correct preceding ciphertext block.
3- **Error propagation**: a single bit error in ciphertext block cj affects decipherment of blocks cj and cj+1 (since xj depends on cj and cj−1).
4- **Error recovery**: the CBC mode is *self-synchronizing* or *ciphertext autokey* in the sense that if an error (including loss of one or more entire blocks) occurs in block cj but not cj+1, cj+2 is correctly decrypted to xj+2.

- It is possible to convert a *block cipher* into a *stream cipher*, using one of the three modes:
  - **cipher feedback (CFB) mode,**
  - **output feedback (OFB) mode,**
  - **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.

Cipher feedback (CFB) mode

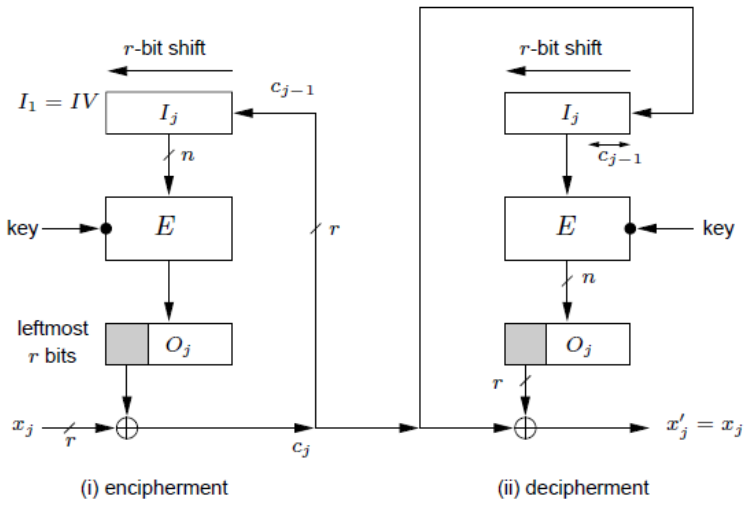
In some applications require that r-bit plaintext units be encrypted and transmitted without delay, for some fixed r < n (often r = 1 or r = 8).

Algorithm CFB mode of operation (CFB-r)

INPUT: k-bit key K; n-bit IV; r-bit plaintext blocks $x_1, \ldots, x_u$ $(1 \leq r \leq n)$.
SUMMARY: produce r-bit ciphertext blocks $c_1, \ldots, c_u$; decrypt to recover plaintext.

1. Encryption: $I_1 \leftarrow IV$. (I_j is the input value in a shift register.) For $1 \leq j \leq u$:
   (a) $O_j \leftarrow E_K(I_j)$. (Compute the block cipher output.)
   (b) $t_j \leftarrow$ the r leftmost bits of O_j. (Assume the leftmost is identified as bit 1.)
   (c) $c_j \leftarrow x_j \oplus t_j$. (Transmit the r-bit ciphertext block c_j.)
   (d) $I_{j+1} \leftarrow 2^r \cdot I_j + c_j \bmod 2^n$. (Shift c_j into right end of shift register.)
2. Decryption: $I_1 \leftarrow IV$. For $1 \leq j \leq u$, upon receiving c_j:
   $x_j \leftarrow c_j \oplus t_j$, where t_j, O_j and I_j are computed as above.

c) Cipher feedback (CFB), r-bit characters/r-bit feedback



Properties of the CFB mode of operation:

1- **Identical plaintexts**: as per CBC encryption, changing the IV results in the same plaintext input being enciphered to a different output.
2- **Chaining dependencies**: similar to CBC encryption, the chaining mechanism causes ciphertext block cj to depend on both xj and preceding plaintext blocks; consequently, re-ordering ciphertext blocks affects decryption. Proper decryption of a correct ciphertext block requires the preceding n/r ciphertext blocks to be correct.
3- **Error propagation**: one or more bit errors in any single r-bit ciphertext block cj affects the decipherment of that and the next n/r ciphertext blocks. The recovered plaintext xj will differ from xj precisely in the bit positions cj was in

error; the other incorrectly recovered plaintext blocks will typically be random vectors, i.e., have 50% of bits in error.

4- **Error recovery**: the CFB mode is self-synchronizing similar to CBC, but requires n/r ciphertext blocks to recover.
5- **Throughput**: for r < n, throughput is decreased by a factor of n/r (vs. CBC) in that each execution of E yields only r bits of ciphertext output.

Note: encryption function E is used for both CFB encryption and decryption

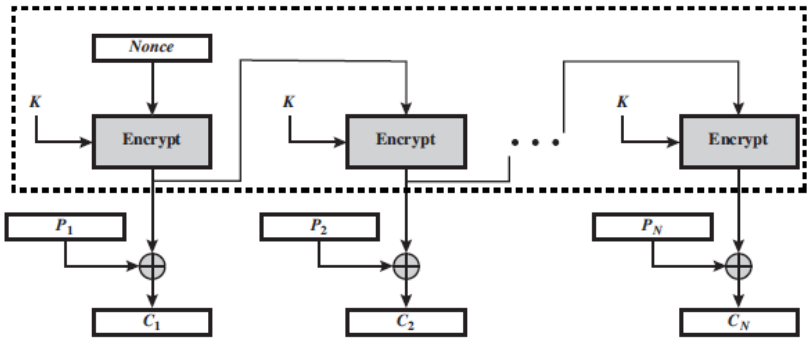
Output feedback (OFB) mode

This mode is used for applications in which all error propagation must be avoided

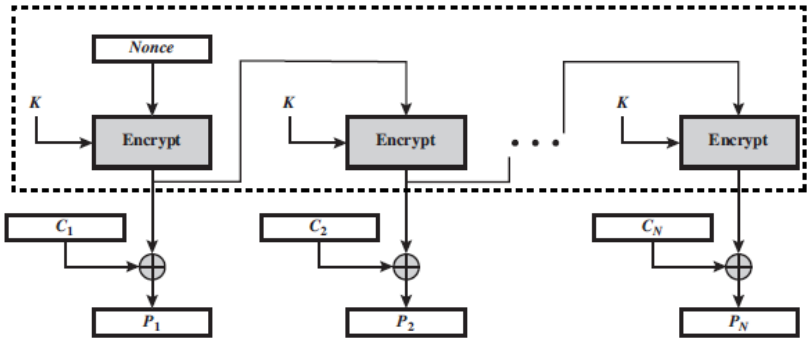
Algorithm OFB mode with full feedback (per ISO 10116)

INPUT: k-bit key K; n-bit IV; r-bit plaintext blocks $x_1, \ldots, x_u$ $(1 \leq r \leq n)$.
SUMMARY: produce r-bit ciphertext blocks $c_1, \ldots, c_u$; decrypt to recover plaintext.

1. Encryption: $I_1 \leftarrow IV$. For $1 \leq j \leq u$, given plaintext block x_j:
   (a) $O_j \leftarrow E_K(I_j)$. (Compute the block cipher output.)
   (b) $t_j \leftarrow$ the r leftmost bits of O_j. (Assume the leftmost is identified as bit 1.)
   (c) $c_j \leftarrow x_j \oplus t_j$. (Transmit the r-bit ciphertext block c_j.)
   (d) $I_{j+1} \leftarrow O_j$. (Update the block cipher input for the next block.)
2. Decryption: $I_1 \leftarrow IV$. For $1 \leq j \leq u$, upon receiving c_j:
   $x_j \leftarrow c_j \oplus t_j$, where t_j, O_j, and I_j are computed as above.



(a) Encryption



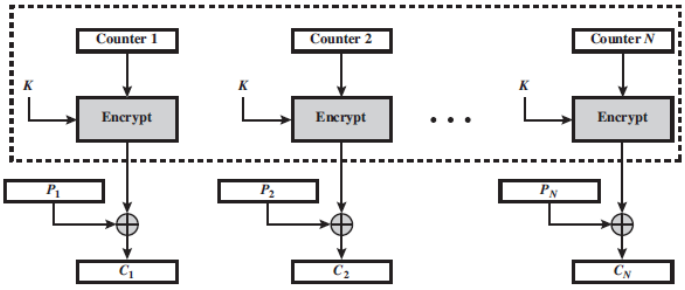
(b) Decryption

Properties of the OFB mode of operation:

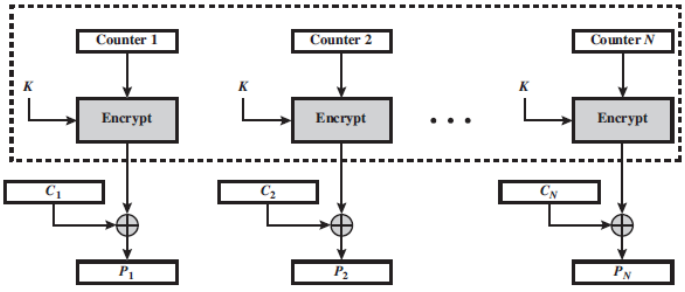
1- **Identical plaintexts**: as per CBC andCFB modes, changing the IV results in the same plaintext being enciphered to a different output.
2- **Chaining dependencies**: the keystream is plaintext-independent.
3- **Error propagation**: one or more bit errors in any ciphertext character cj affects the decipherment of only that character, in the precise bit position(s) cj is in error, causing the corresponding recovered plaintext bit(s) to be complemented.
4- **Error recovery**: the OFB mode recovers from ciphertext bit errors, but cannot self synchronize after loss of ciphertext bits, which destroys alignment of the decrypting keystream.
5- **Throughput**: for r < n, throughput is decreased as per the CFB mode. However, in all cases, since the keystream is independent of plaintext or ciphertext, it may be pre-computed (given the key and IV).

Counter (CTR) mode

A simplification of OFB involves updating the input block as a counter, Ij+1 = Ij +1, rather than using feedback. It allows recovery from errors in computing E. Moreover, it provides a random-access property: ciphertext block i need not be decrypted in order to decrypt block i + 1.



(a) Encryption



(b) Decryption

- Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in **parallel** on multiple blocks of plaintext or ciphertext.
- We can encrypt counter before encryption, since it do not depend on inputs.
- The i^{th} block of plaintext or ciphertext can be processed in random-access fashion.