MODULE-II

SYMMETRIC KEY CIPHER

- Block Cipher principles, DES, AES, Blowfish, RC5, IDEA, Block cipher operation, Stream ciphers, RC4.

ASYMMETRIC KEY CIPHERS

- Principles of public key cryptosystems, RSA algorithm, Elgamal Cryptography, Diffie-Hellman Key Exchange, Knapsack Algorithm.

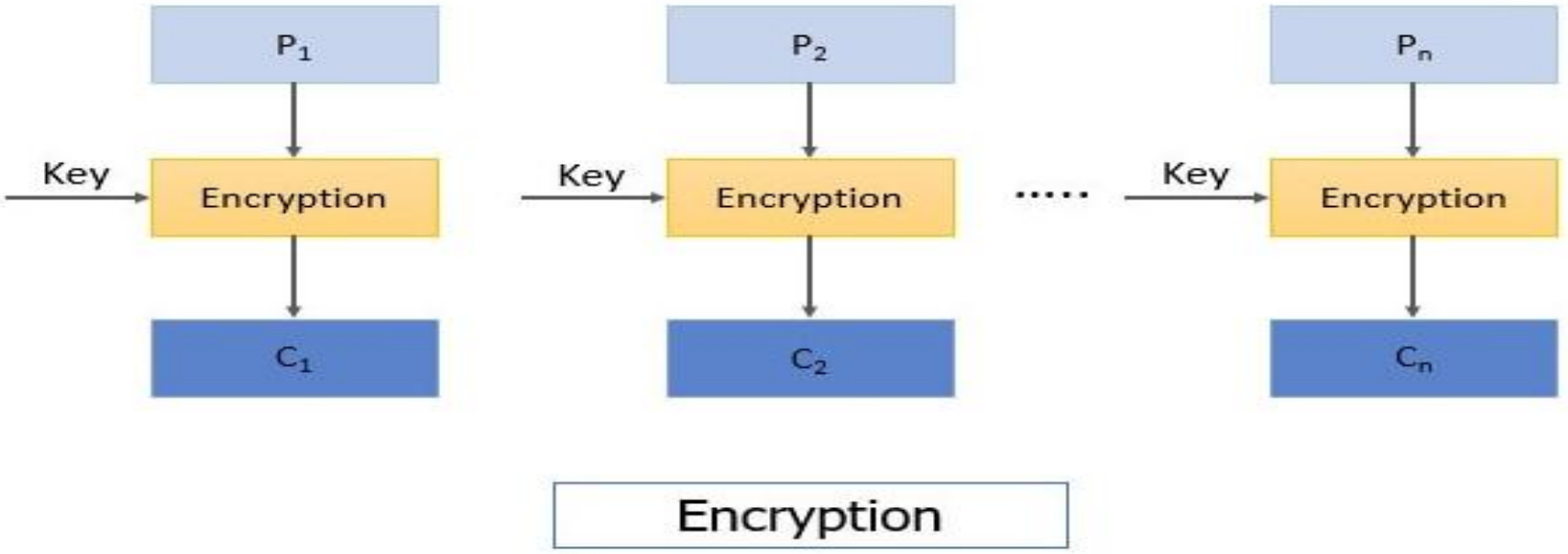
Block Cipher principles

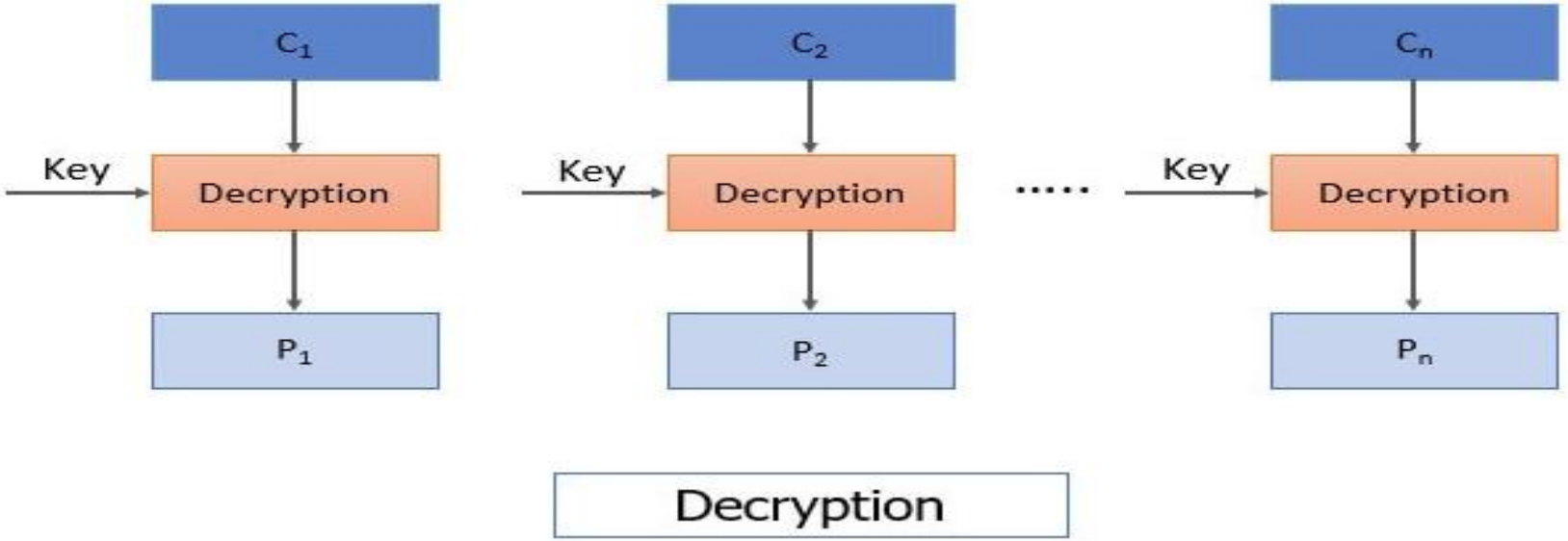
- Block cipher is an encryption method which divides the plain text into blocks of fixed size. Each block has an equal number of bits. At a time, block cipher operates only on one block of plain text and applies key on it to produce the corresponding block of ciphertext.
- **Block Cipher Principles**
  - Number of Rounds
  - Design of Function F
  - Key Schedule Algorithm

Cipher Block modes of Operation

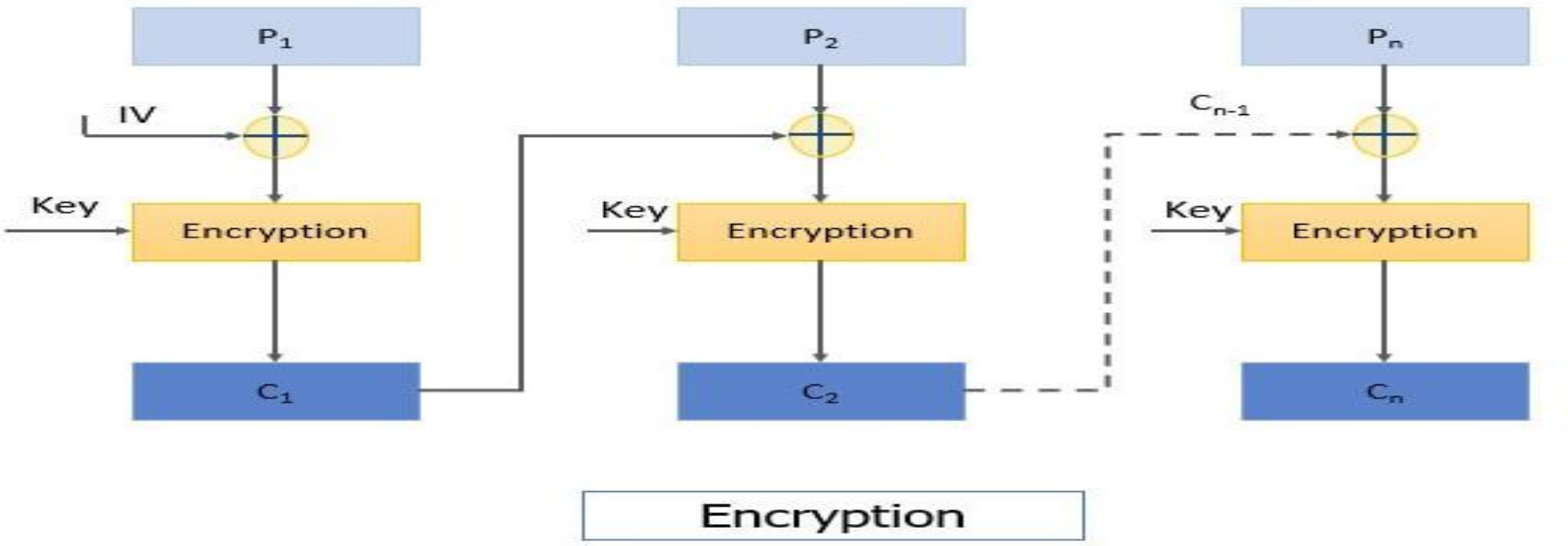
- Electronic Code Book (ECB)
- Cipher Block Chaining Mode (CBC)
- Cipher Feed Back Mode (CFB)
- Output Feedback Mode (OFB)
- Counter Mode (CTR)

Electronic Code Book (ECB)

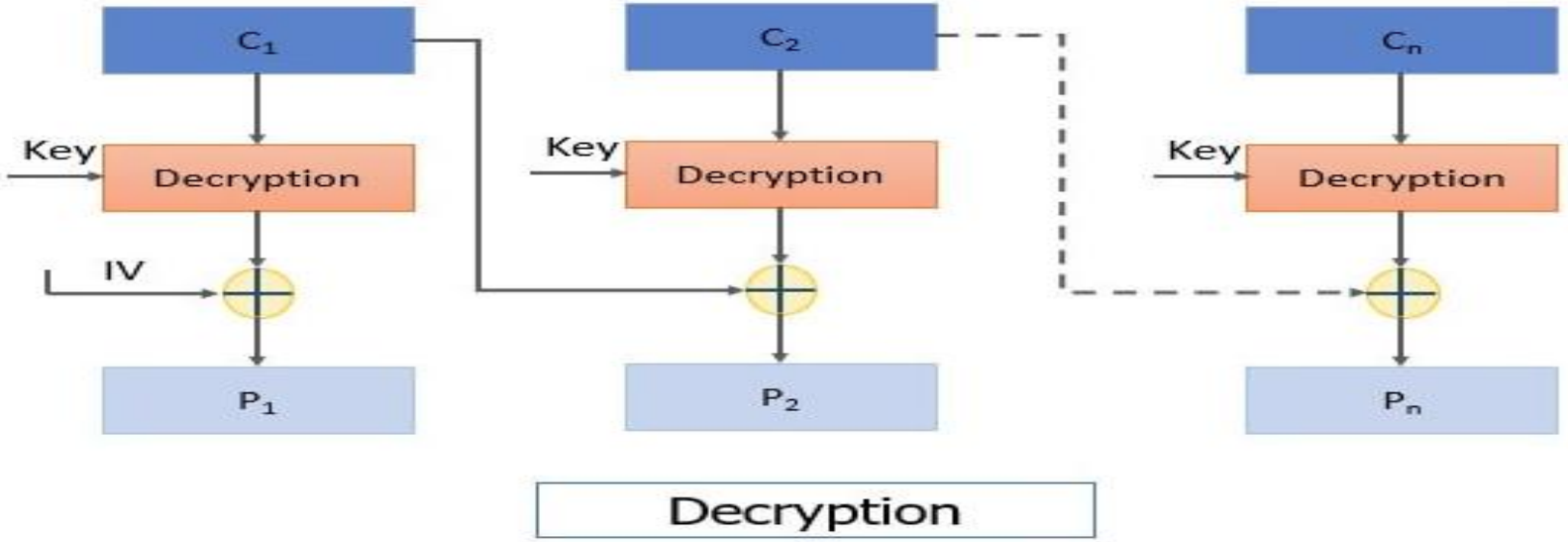




Cipher Block Chaining Mode (CBC)

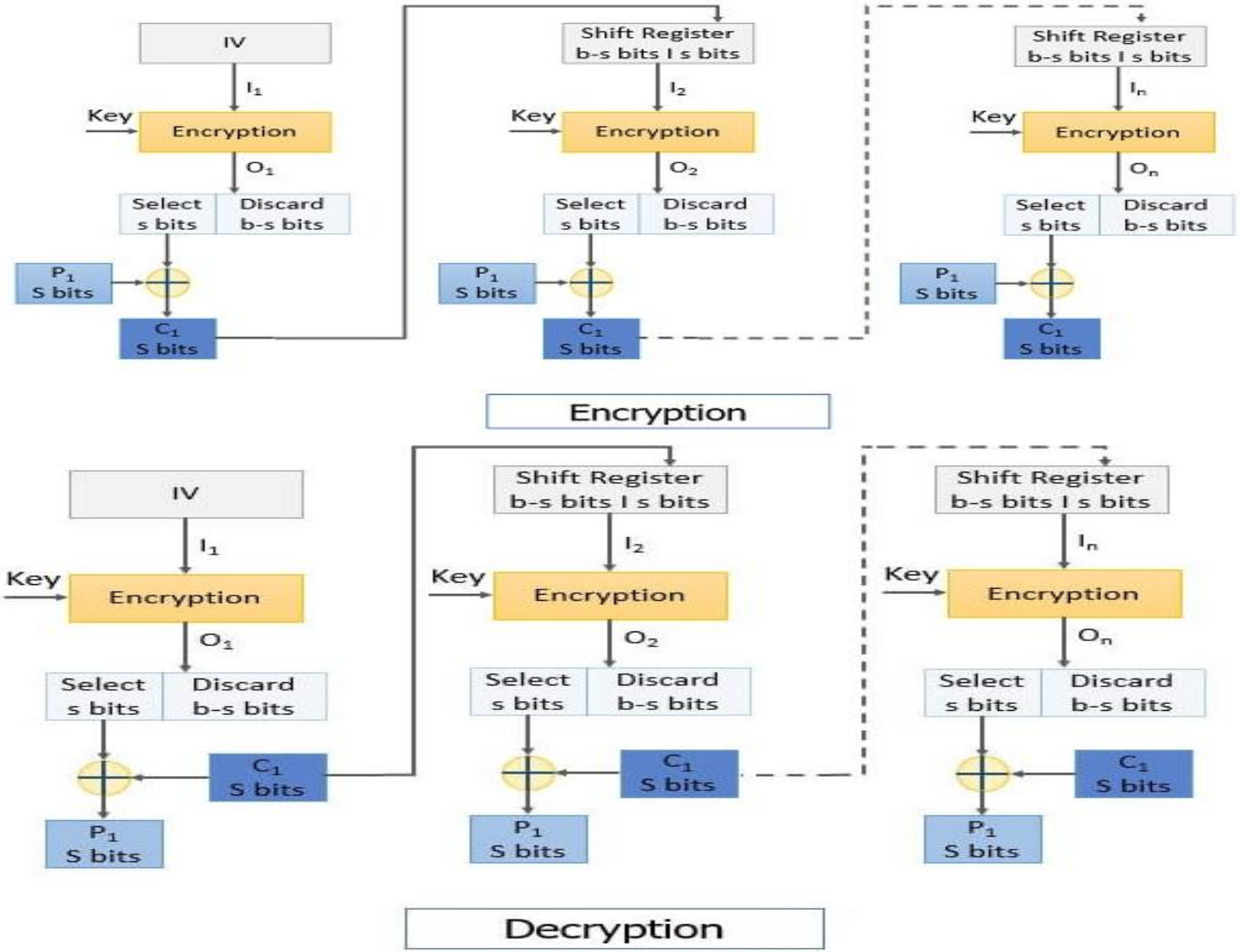


Encryption

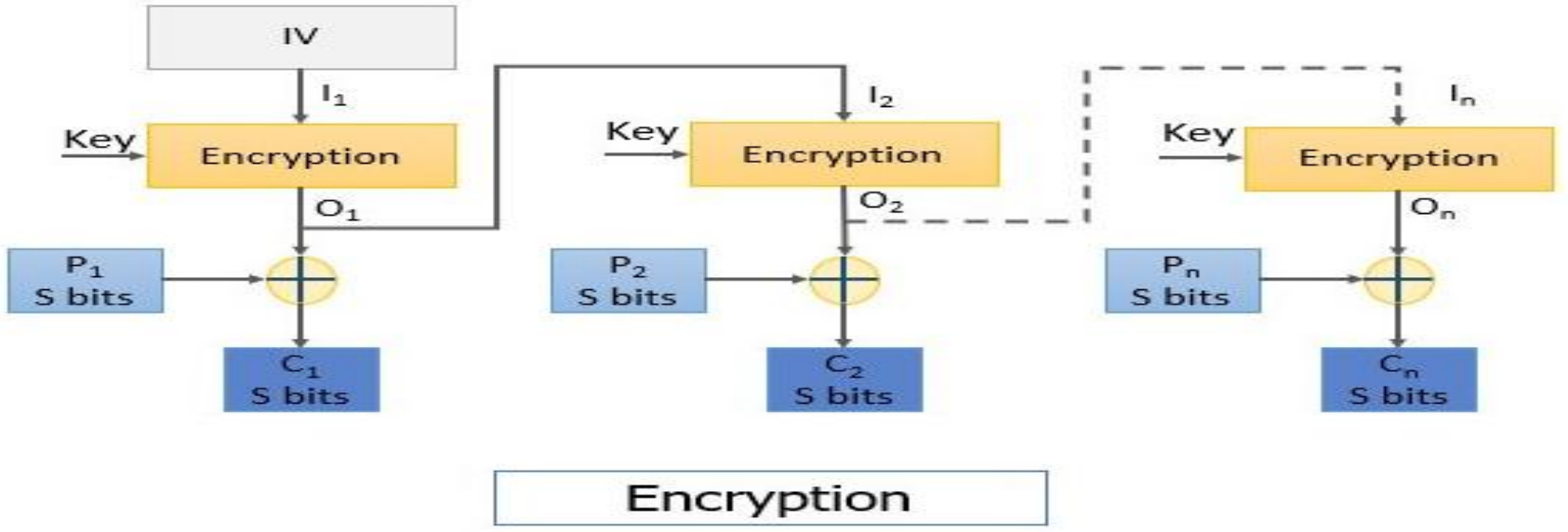


Decryption

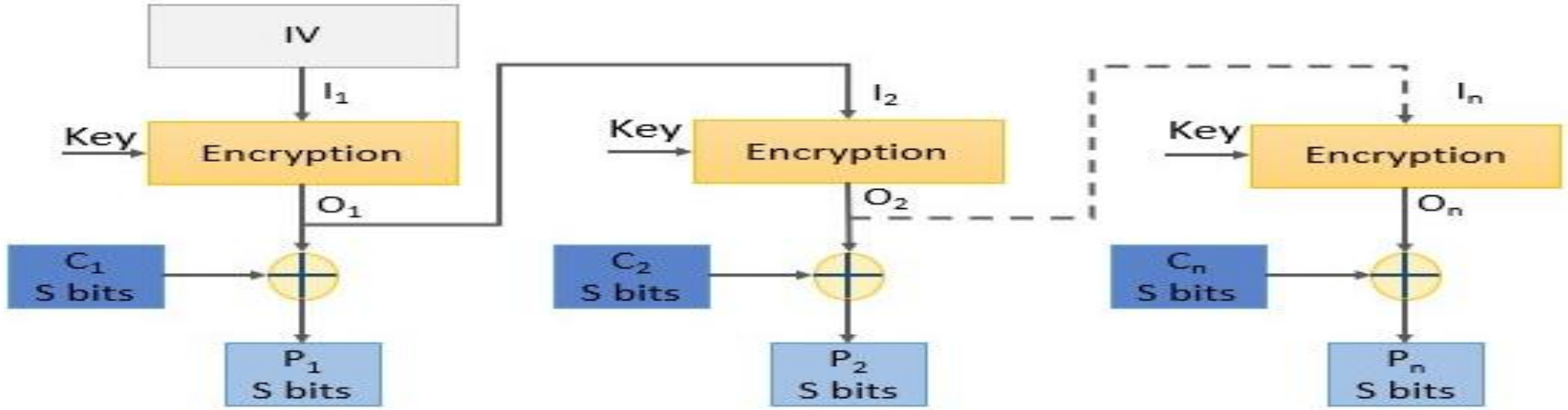
Cipher Feed Back Mode (CFB)



Output Feedback Mode (OFB)

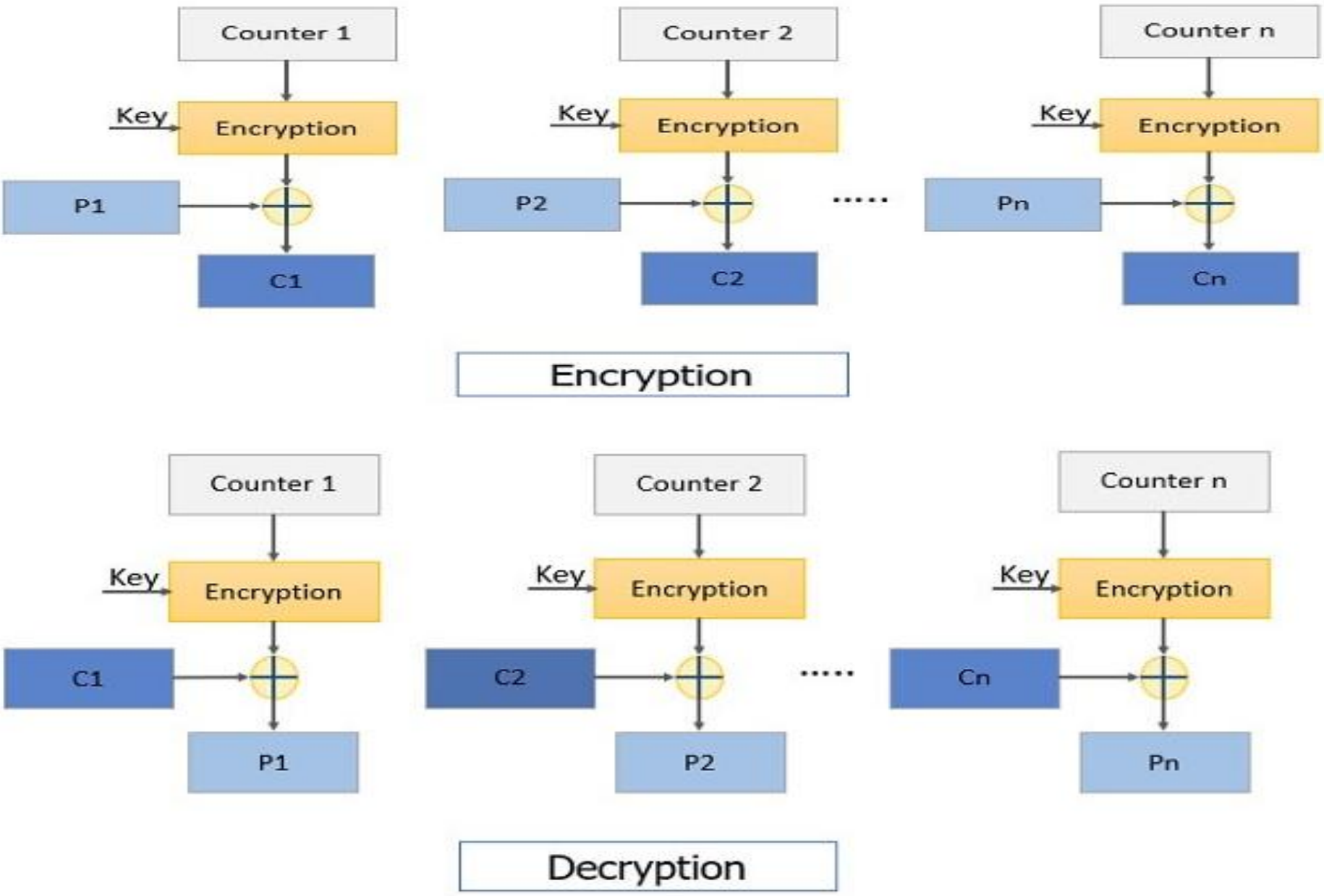


Encryption

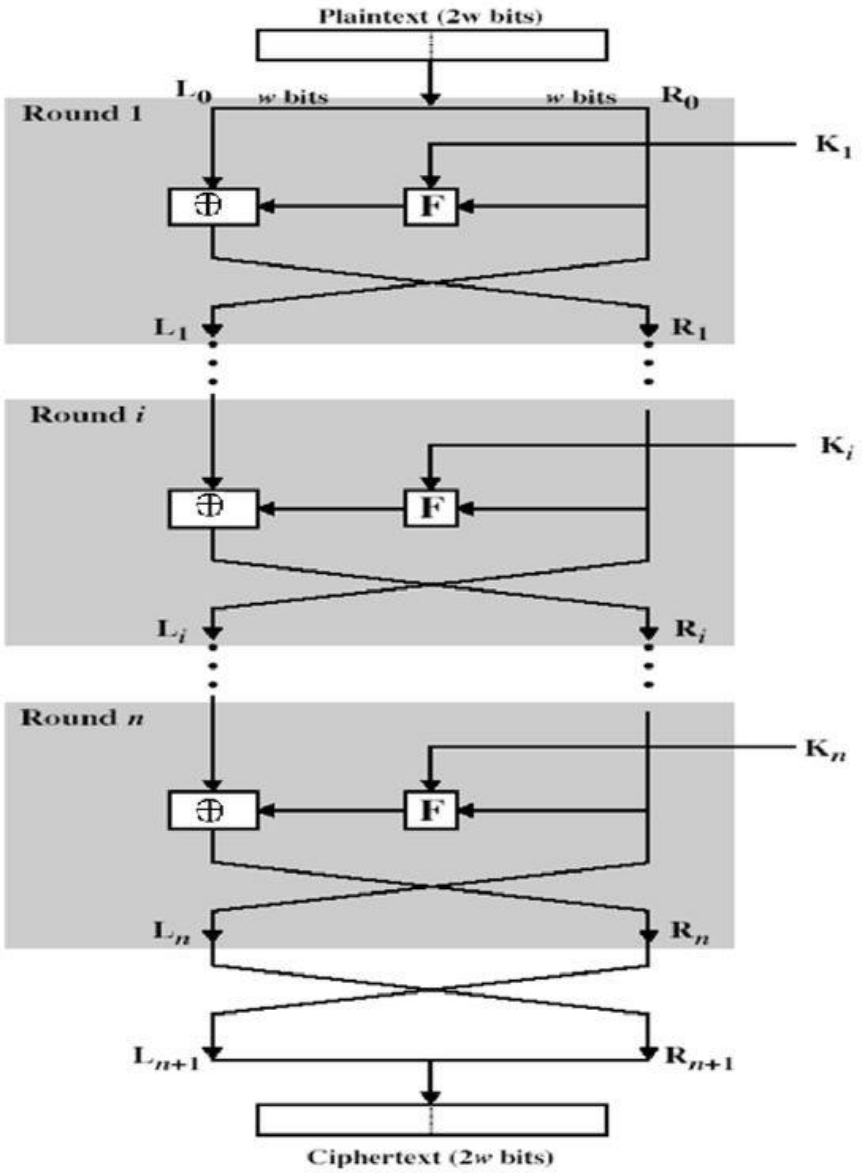


Decryption

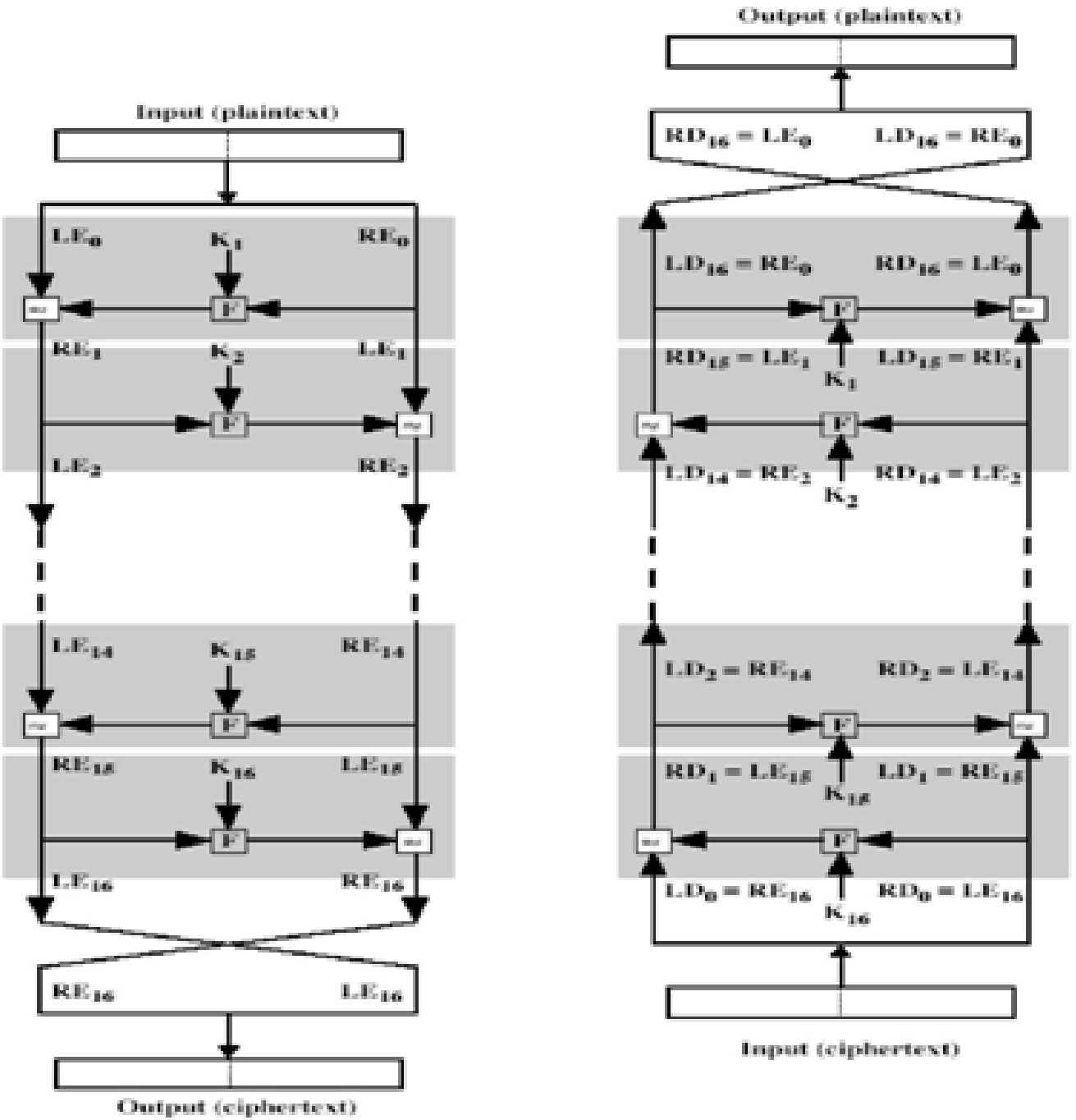
Counter Mode (CTR)

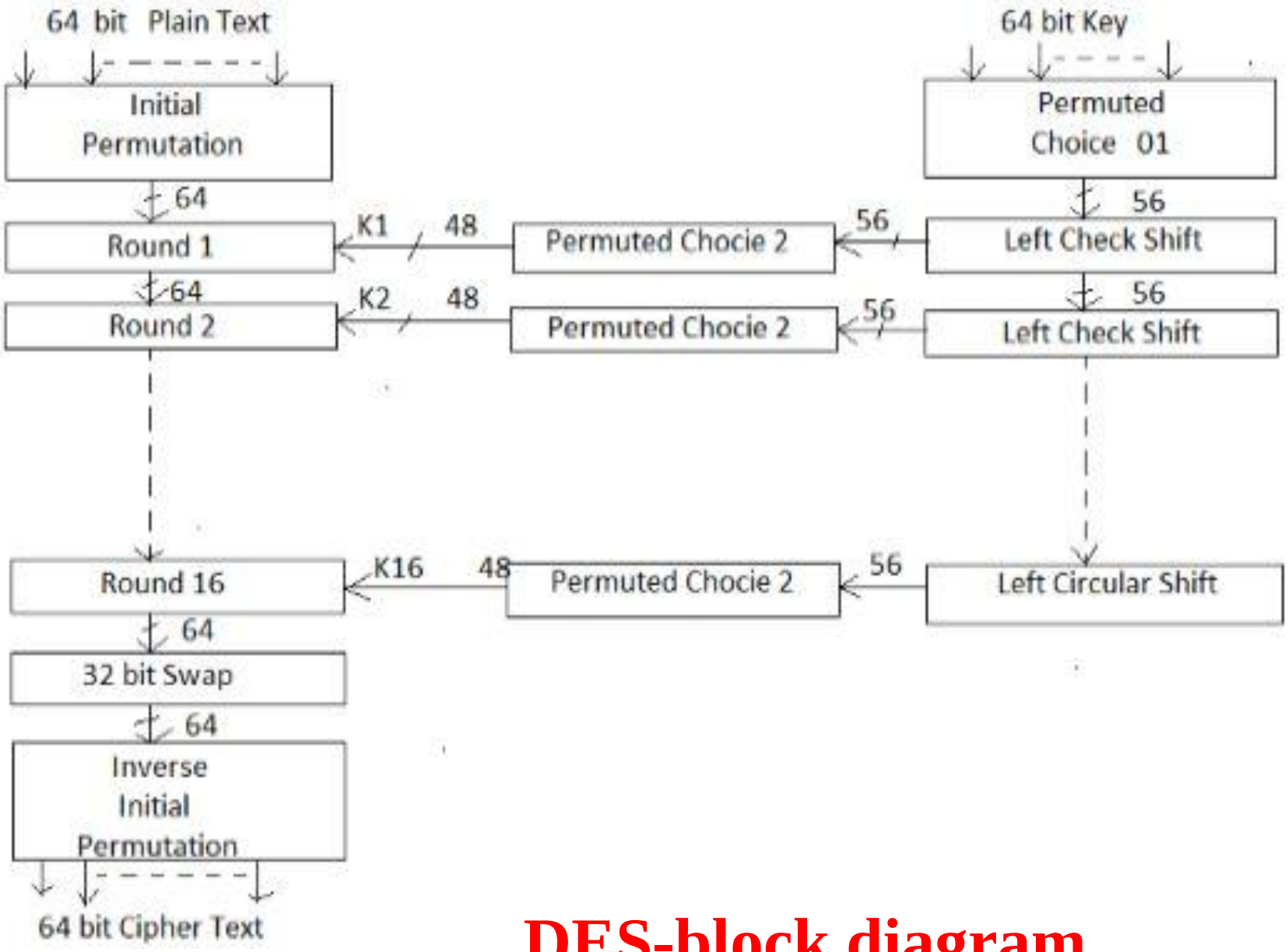


Feistel Cipher Structure

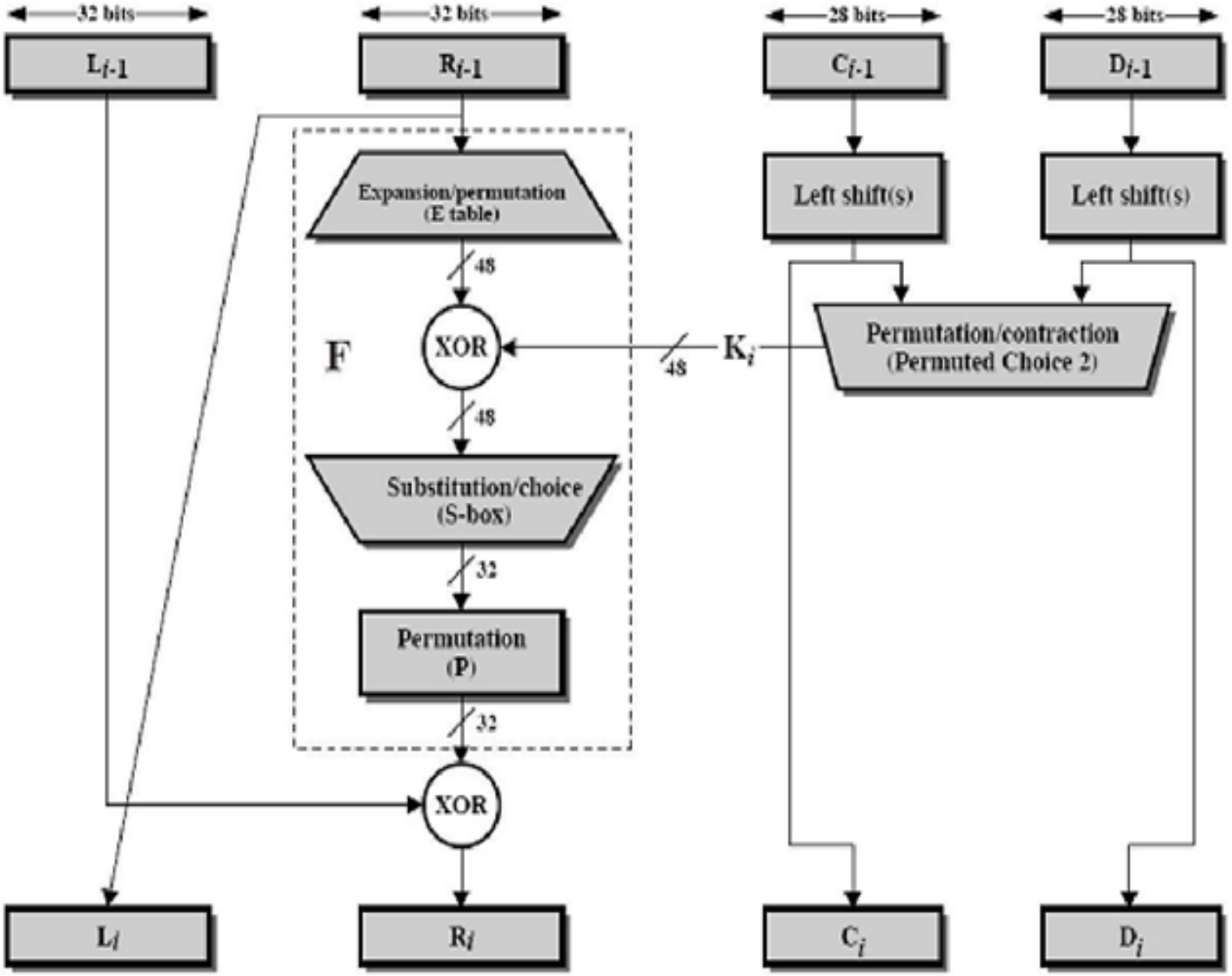


Feistel Cipher Structure

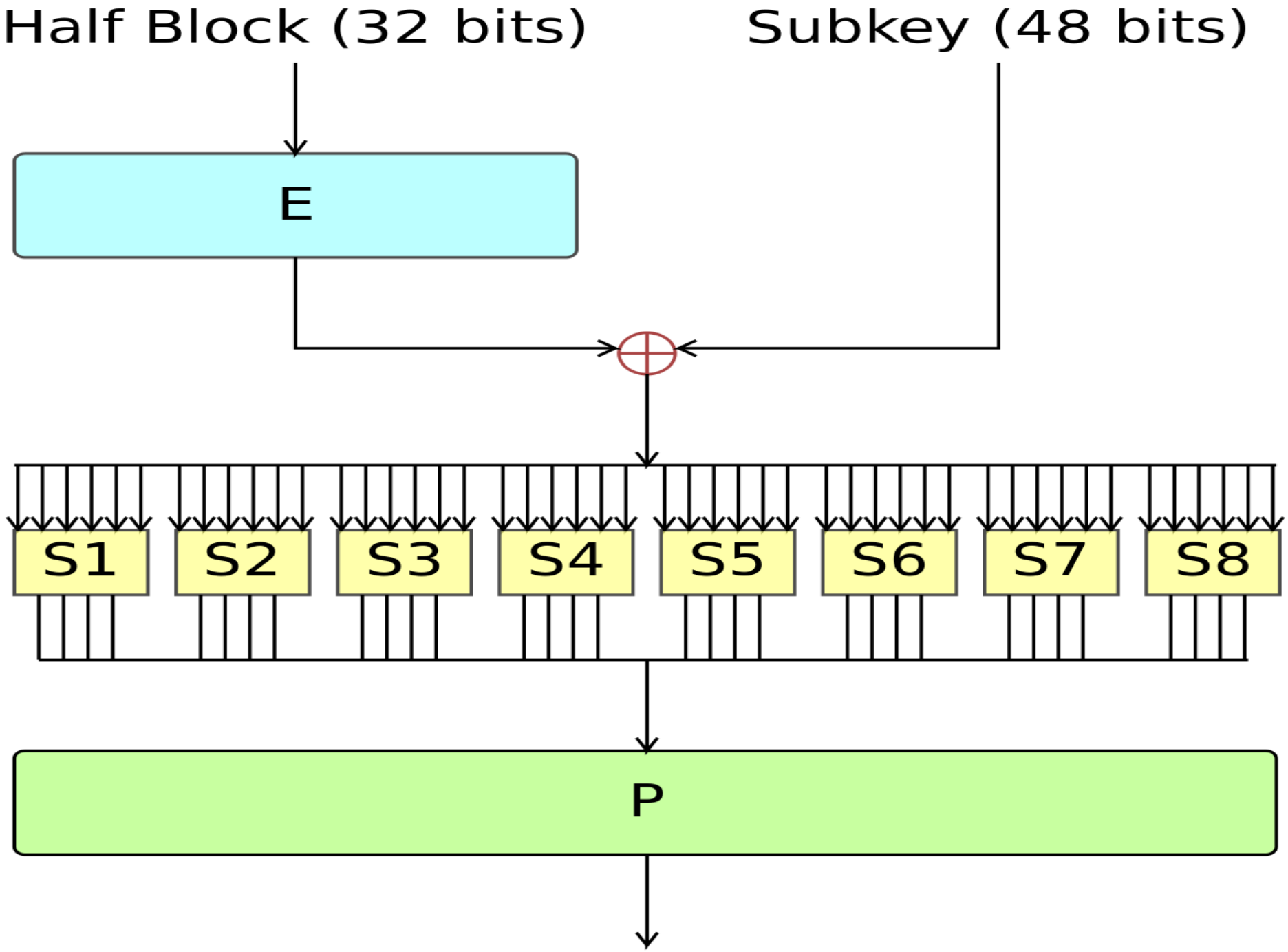

Input (plaintext)
LE_0
K_1
RE_0
F
RE_1
K_2
LE_1
F
LE_2
RE_2
LE_{14}
K_{15}
RE_{14}
F
RE_{15}
K_{16}
LE_{15}
F
LE_{16}
RE_{16}
RE_{16}
LE_{16}
Output (ciphertext)
Output (plaintext)
$RD_{16} = LE_0$
$LD_{16} = RE_0$
$LD_{16} = RE_0$
$RD_{16} = LE_0$
F
$RD_{15} = LE_1$
K_1
$LD_{15} = RE_1$
F
$LD_{14} = RE_2$
K_2
$RD_{14} = LE_2$
$LD_2 = RE_{14}$
$RD_2 = LE_{14}$
F
$RD_1 = LE_{15}$
K_{15}
$LD_1 = RE_{15}$
F
$LD_0 = RE_{16}$
K_{16}
$RD_0 = LE_{16}$
Input (ciphertext)

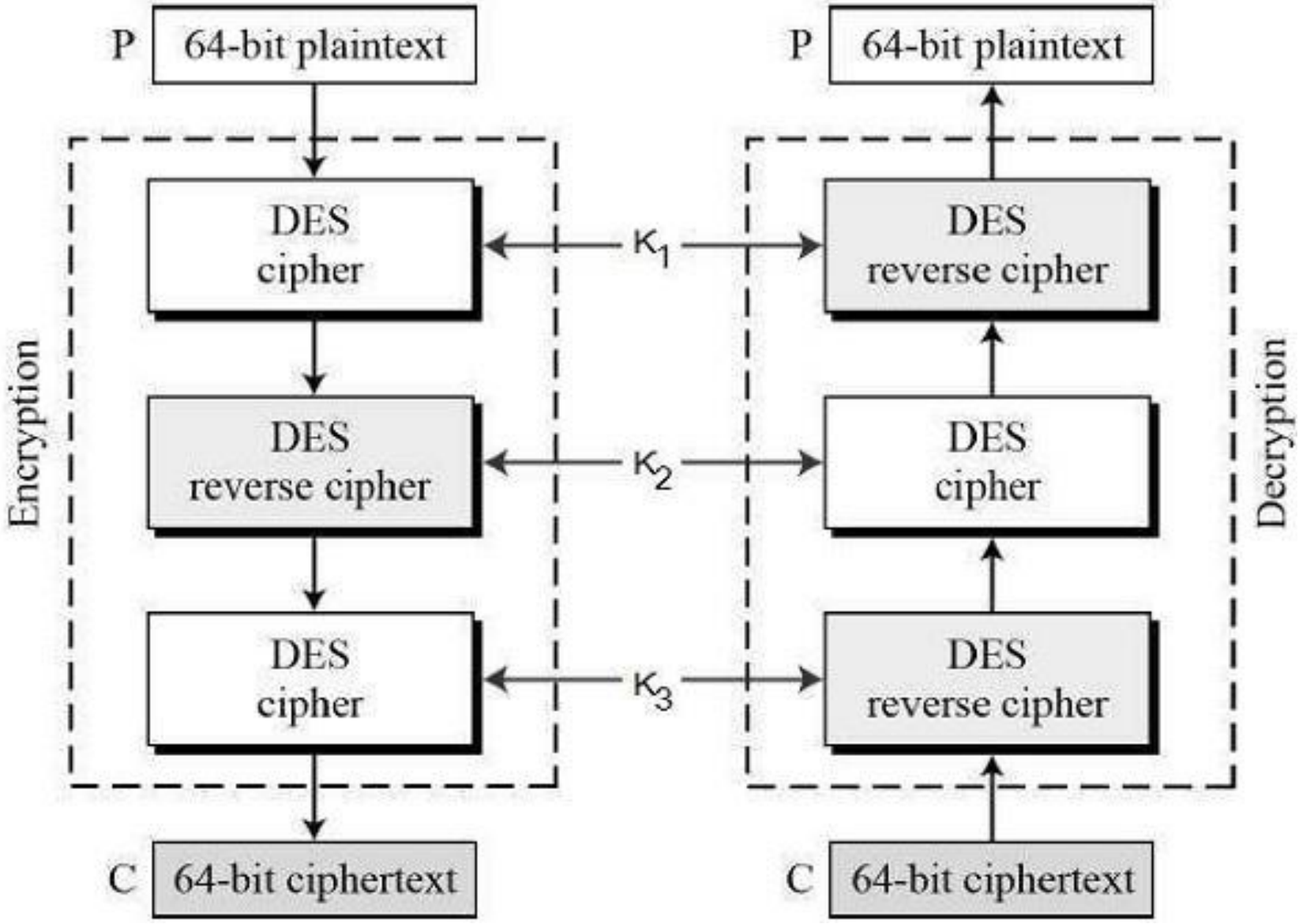


DES-block diagram

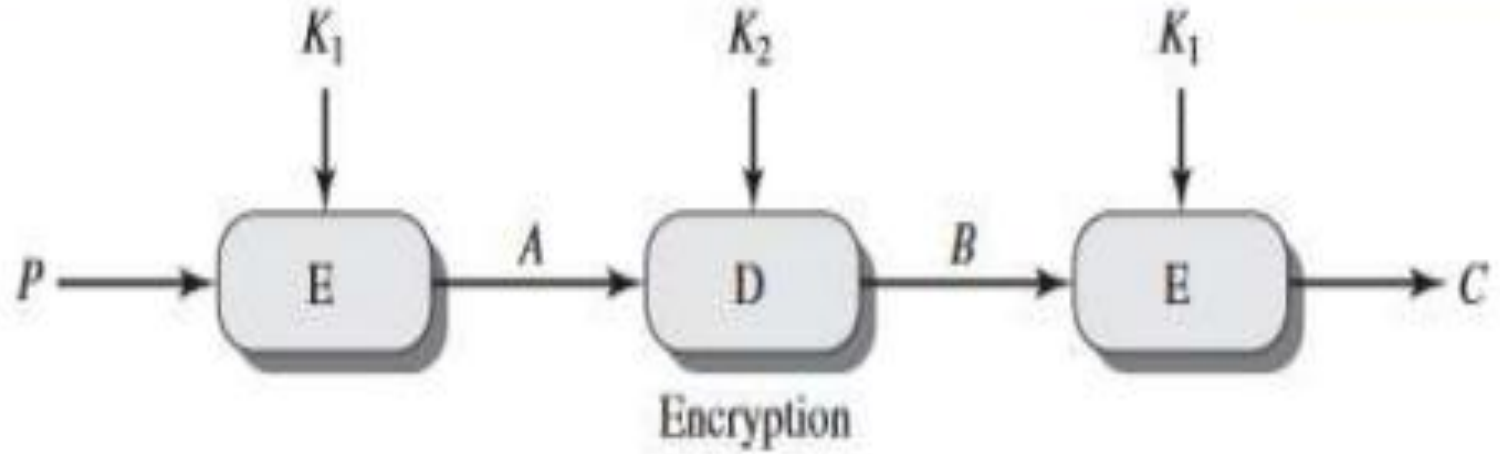


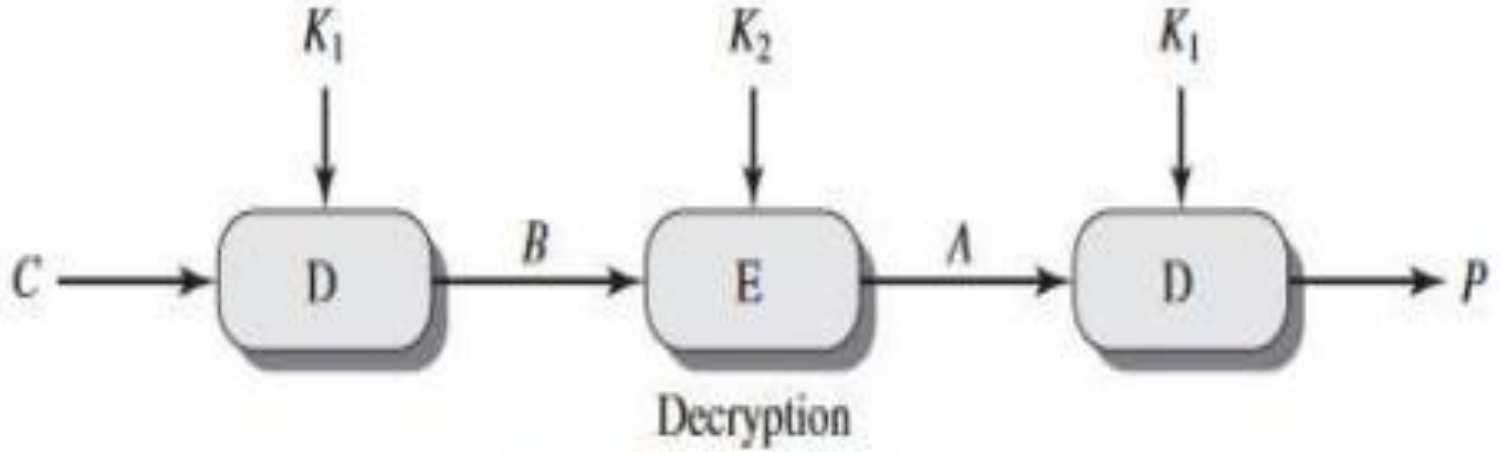
Single Round of DES Algorithm


Half Block (32 bits)
Subkey (48 bits)
E
S1
S2
S3
S4
S5
S6
S7
S8
P

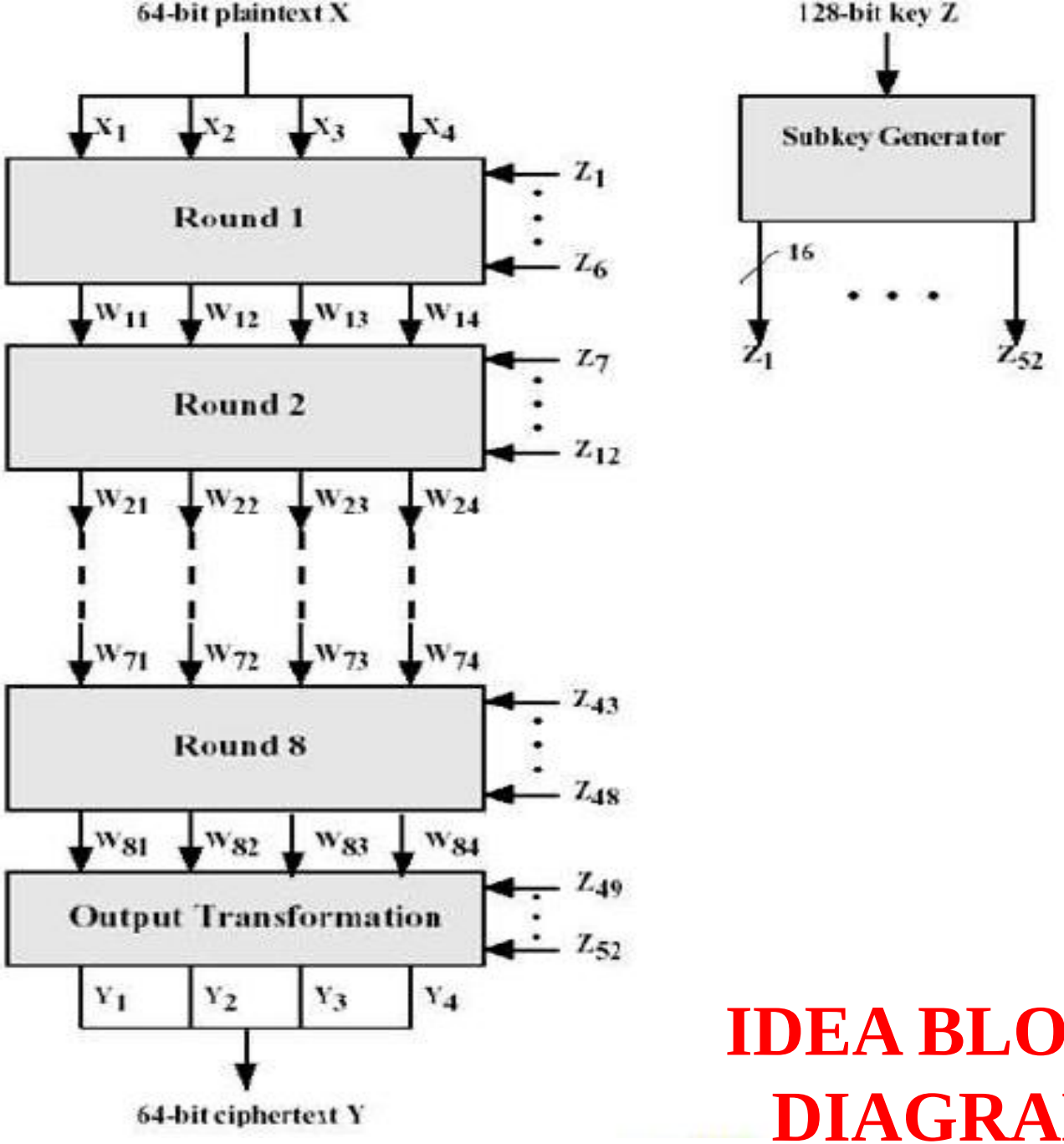

P
64-bit plaintext
DES
cipher
DES
reverse cipher
DES
cipher
C
64-bit ciphertext
Encryption
K1
K2
K3
P
64-bit plaintext
DES
reverse cipher
DES
cipher
DES
reverse cipher
C
64-bit ciphertext
Decryption

Triple DES





(b) Triple encryption



IDEA BLOCK DIAGRAM

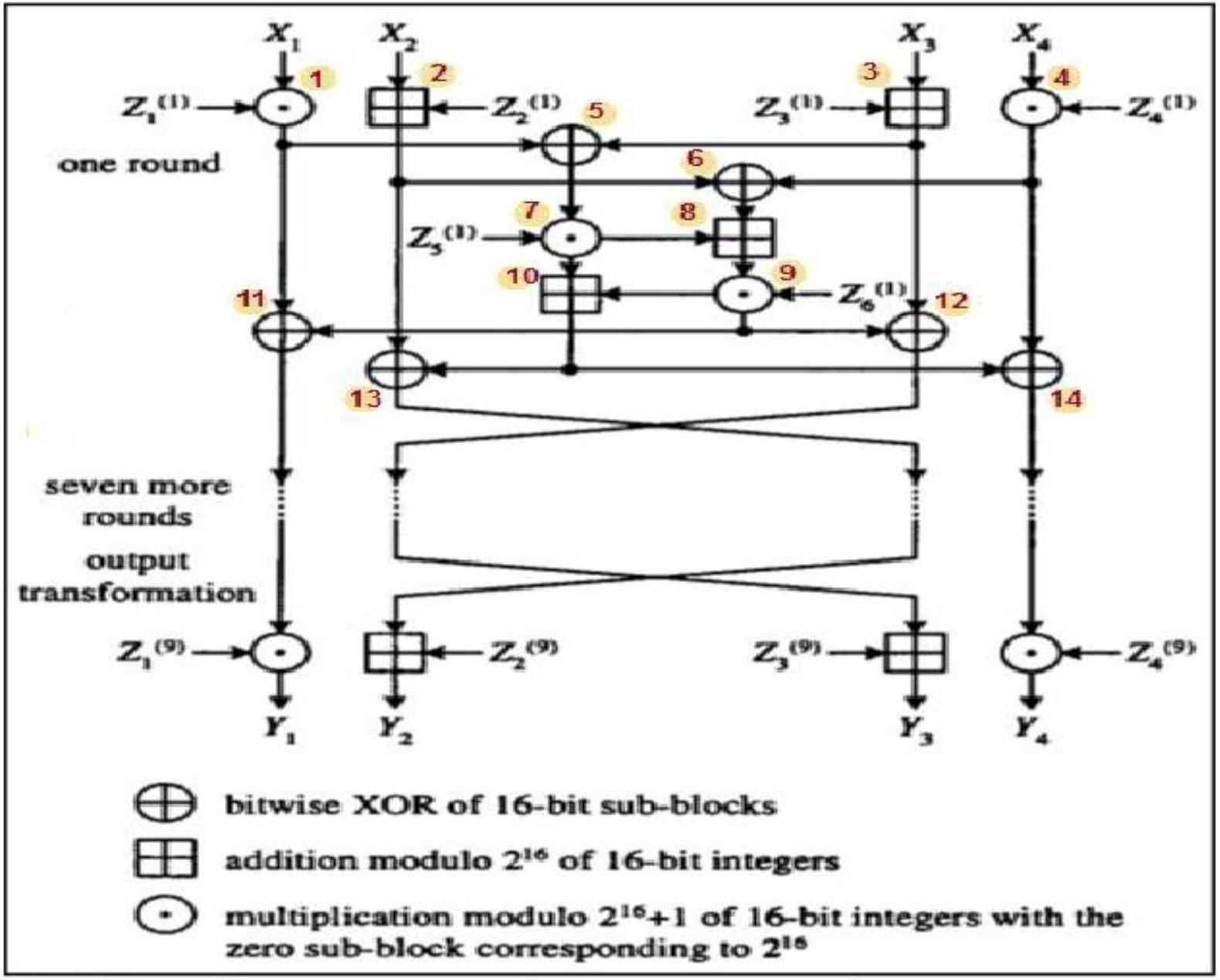
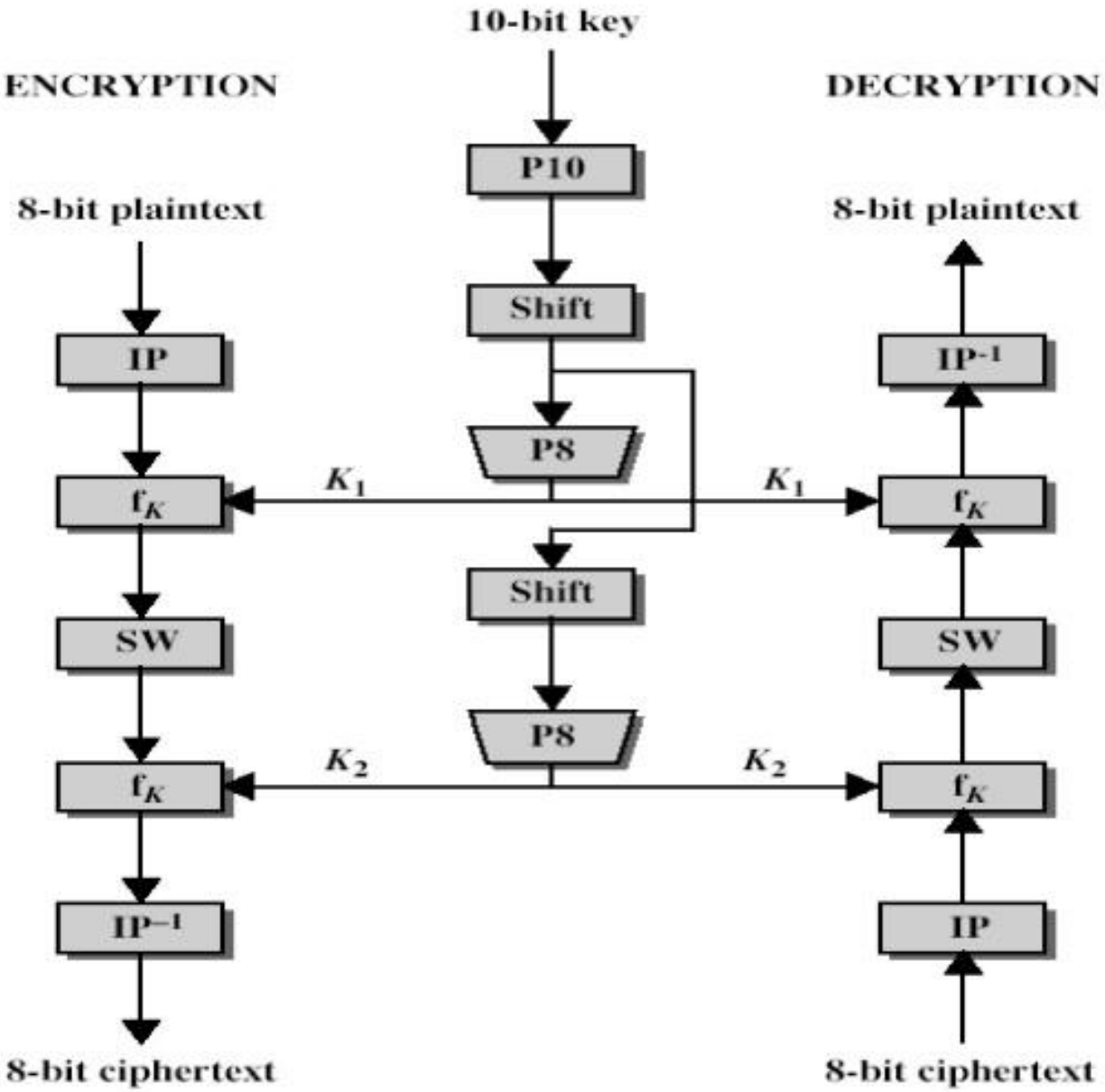


Figure 2: Structure of IDEA encryption process [3]


10-bit key
ENCRYPTION
DECRYPTION
P10
8-bit plaintext
8-bit plaintext
Shift
IP
IP-1
P8
K1
K1
fK
fK
Shift
SW
SW
P8
K2
K2
fK
fK
IP-1
IP
8-bit ciphertext
8-bit ciphertext

S-DES scheme

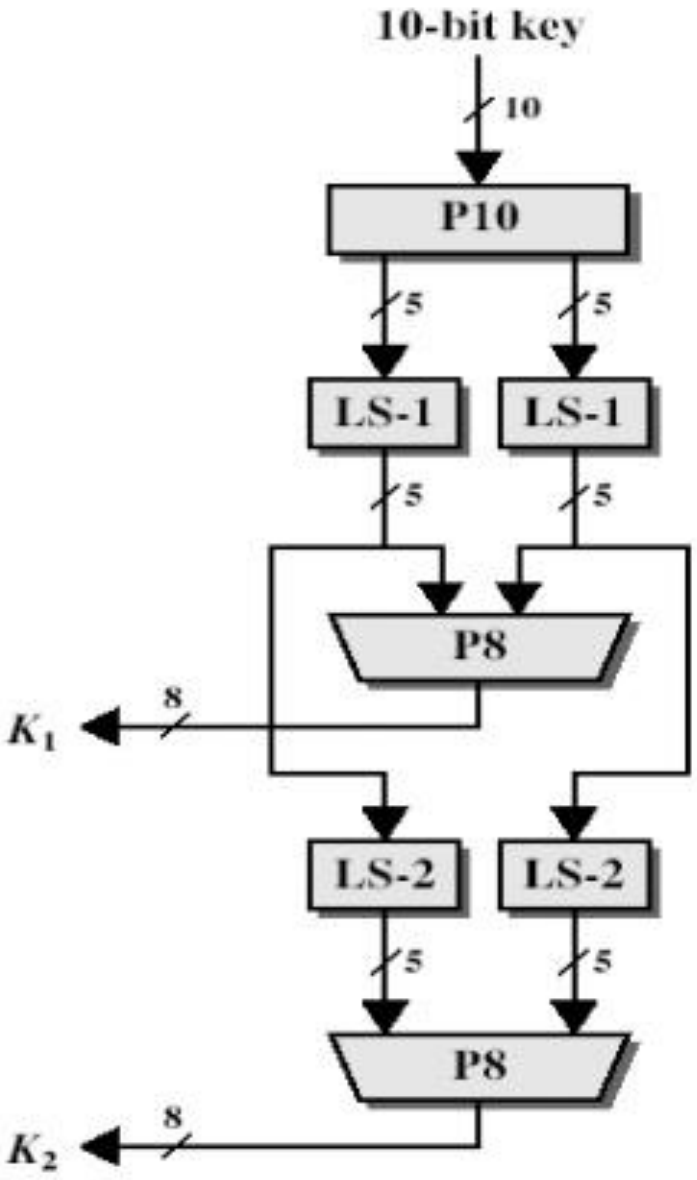
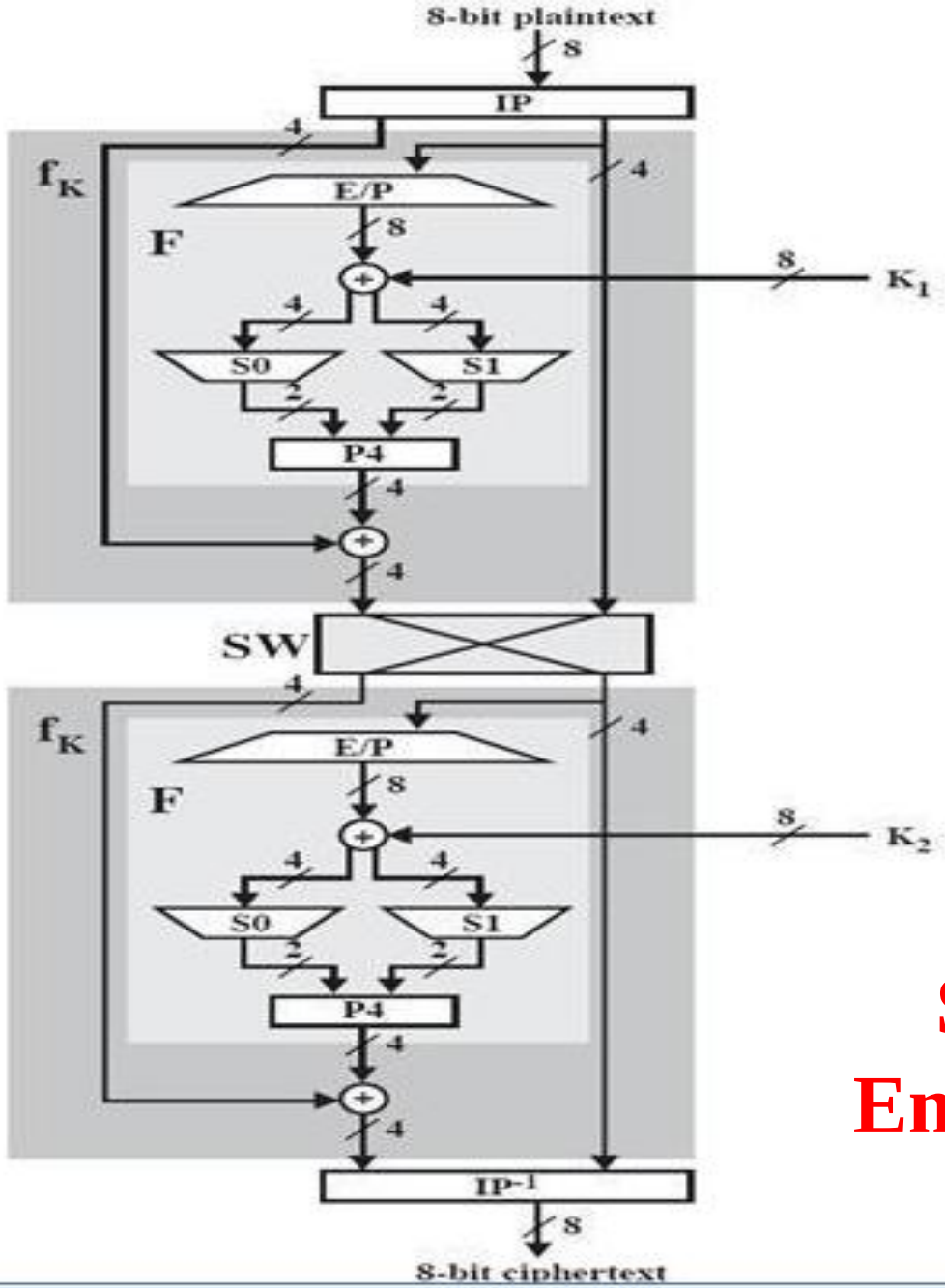
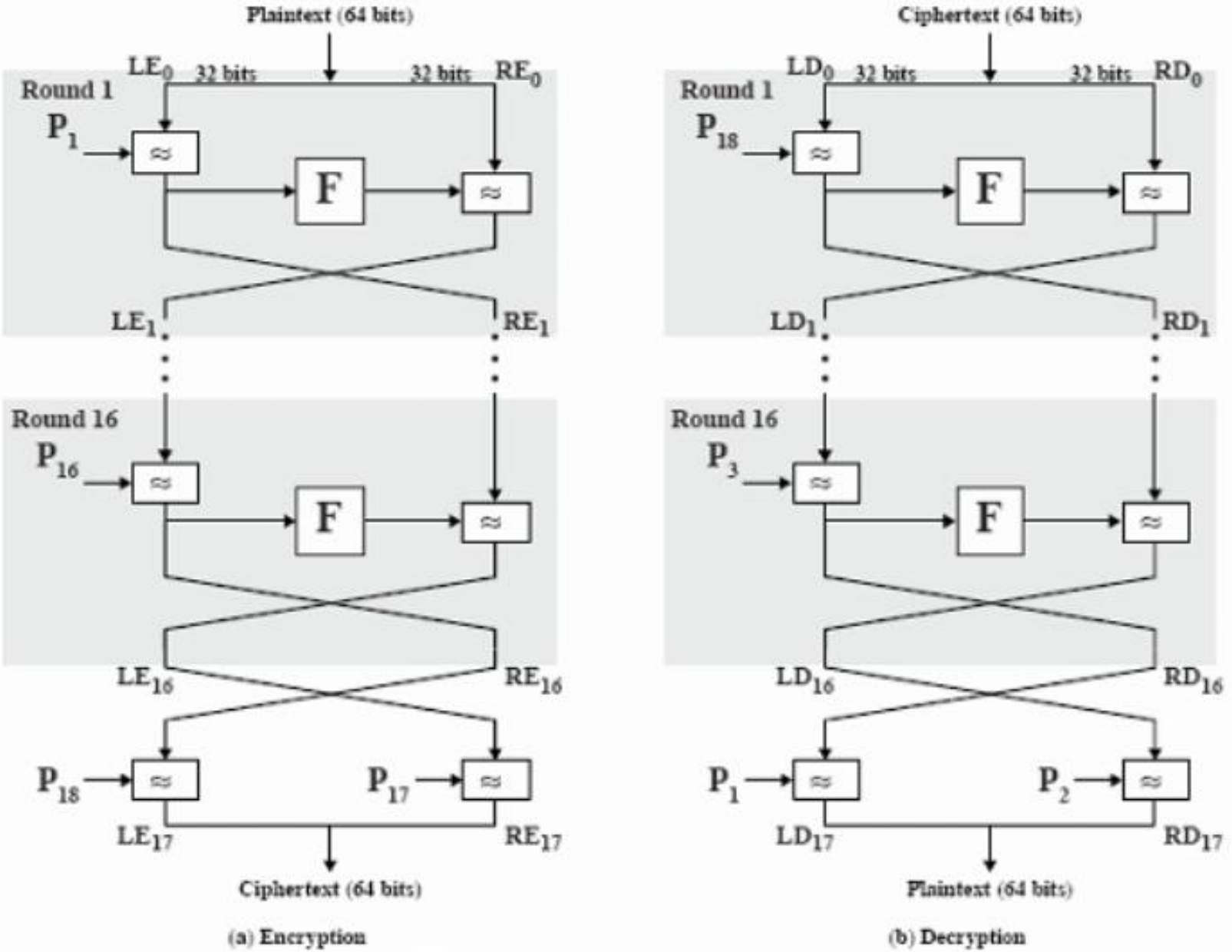


Figure: key generation for S-DES

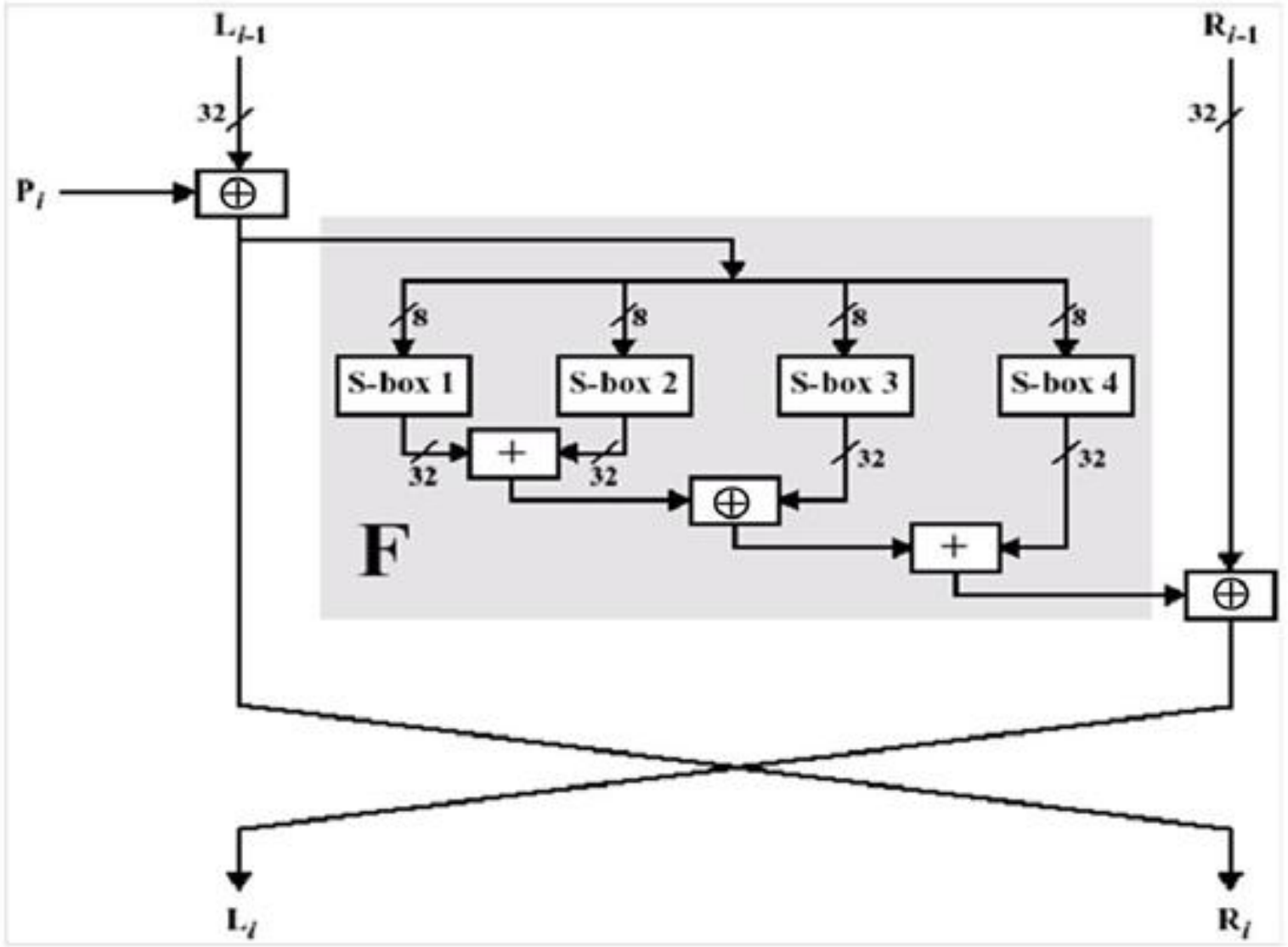


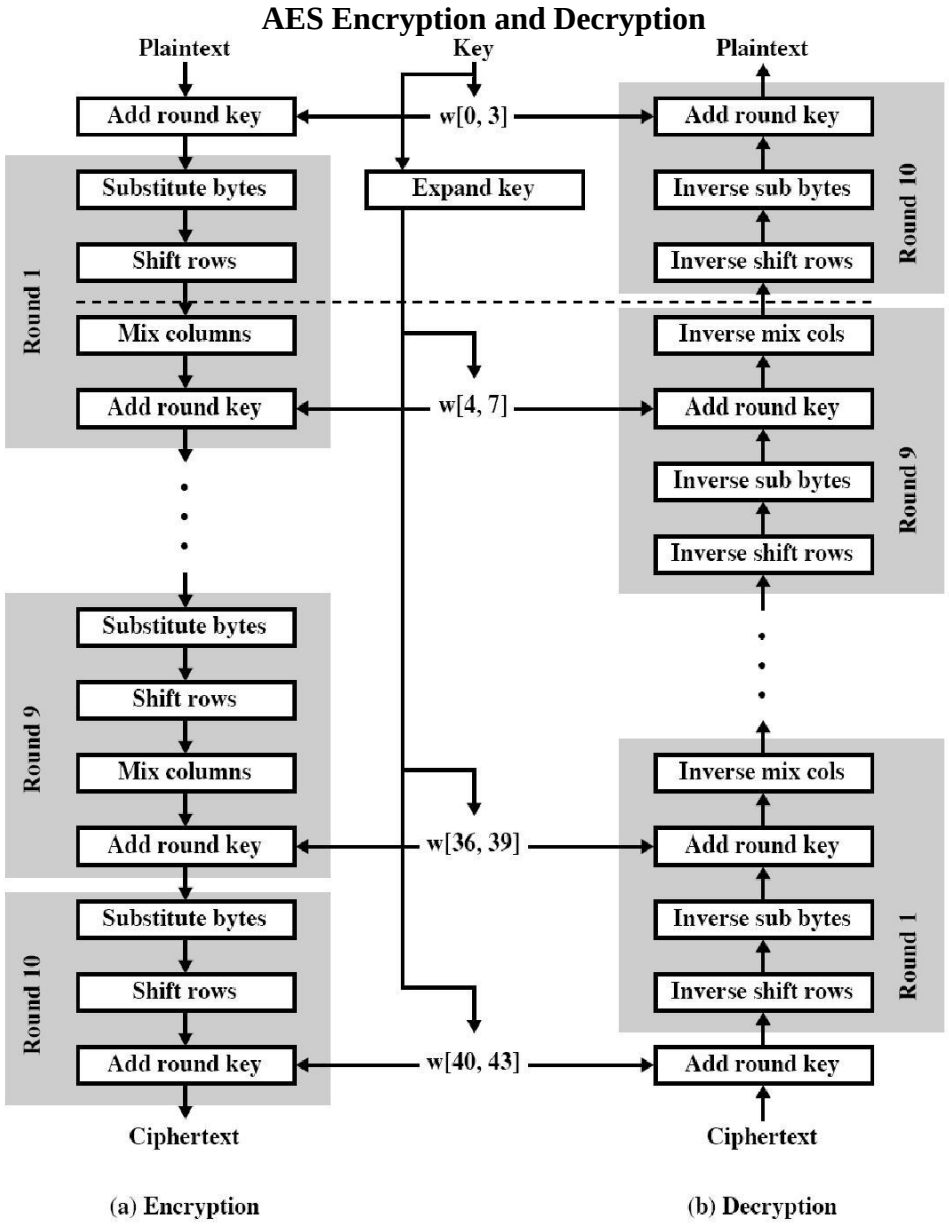
S-DES Encryption

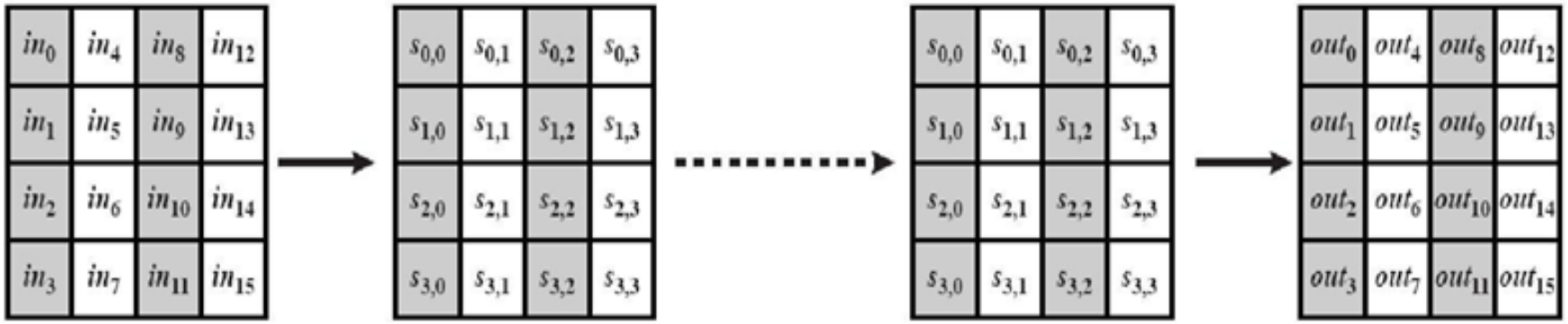


Blowfish Encryption and Decryption

Function



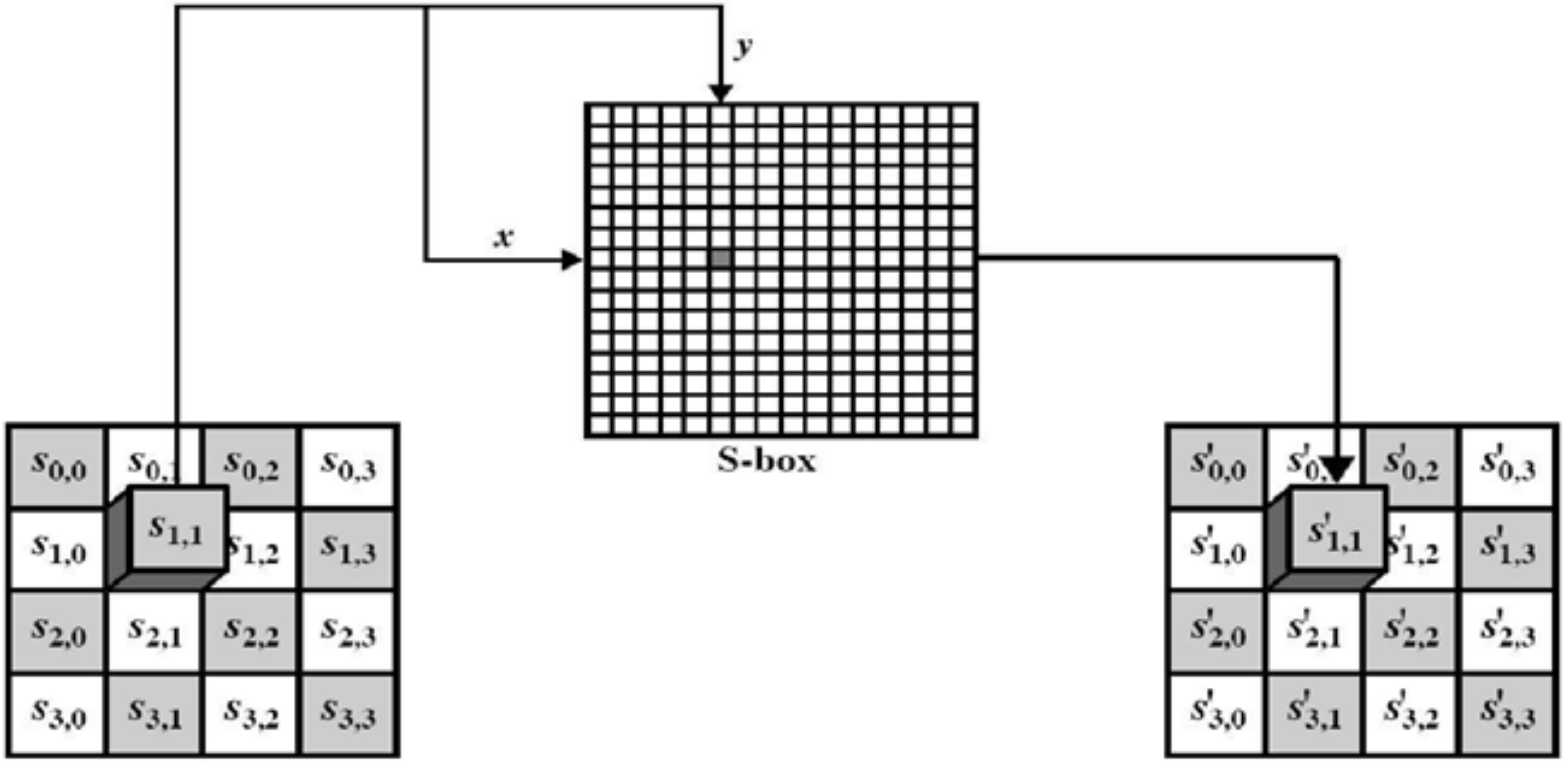

AES Encryption and Decryption
Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Inverse shift rows
Substitute bytes
Shift rows
Inverse mix cols
Mix columns
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Round 10
Shift rows
Inverse shift rows
Round 1
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption



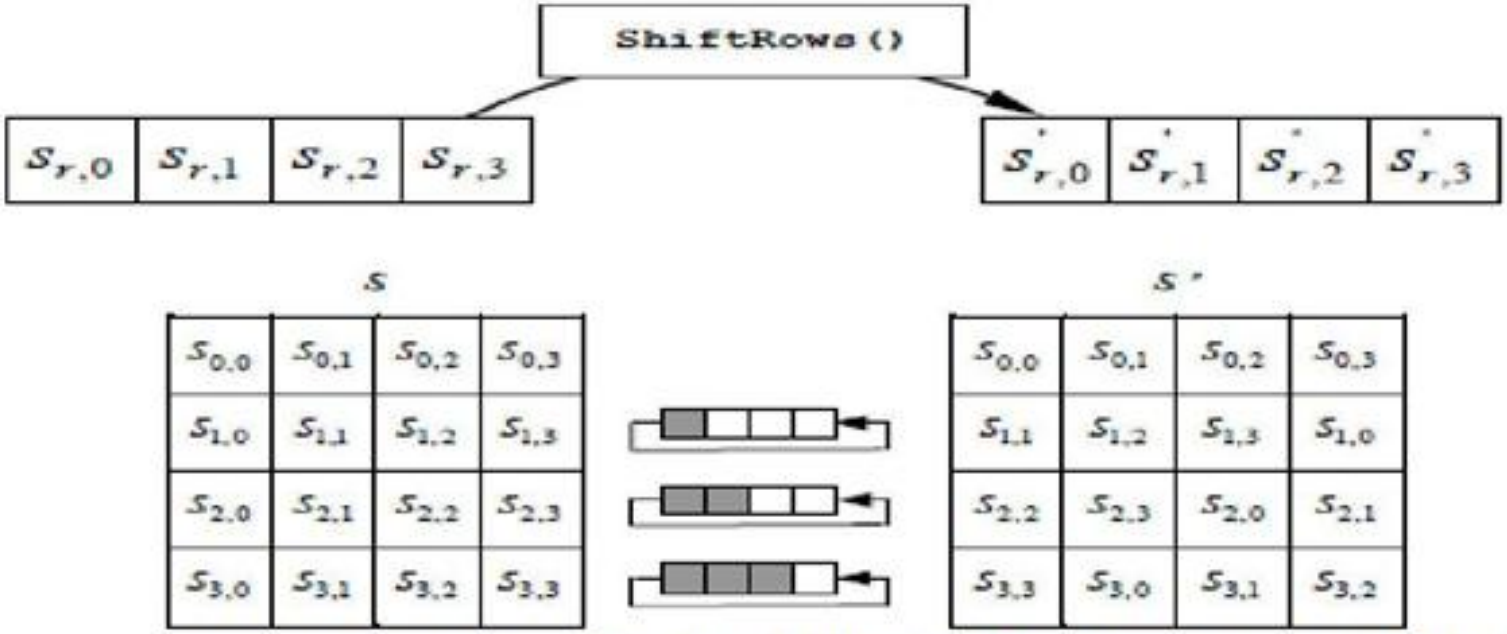
(a) Input, state array, and output

(b) Key and expanded key

Substitute Bytes Transformation

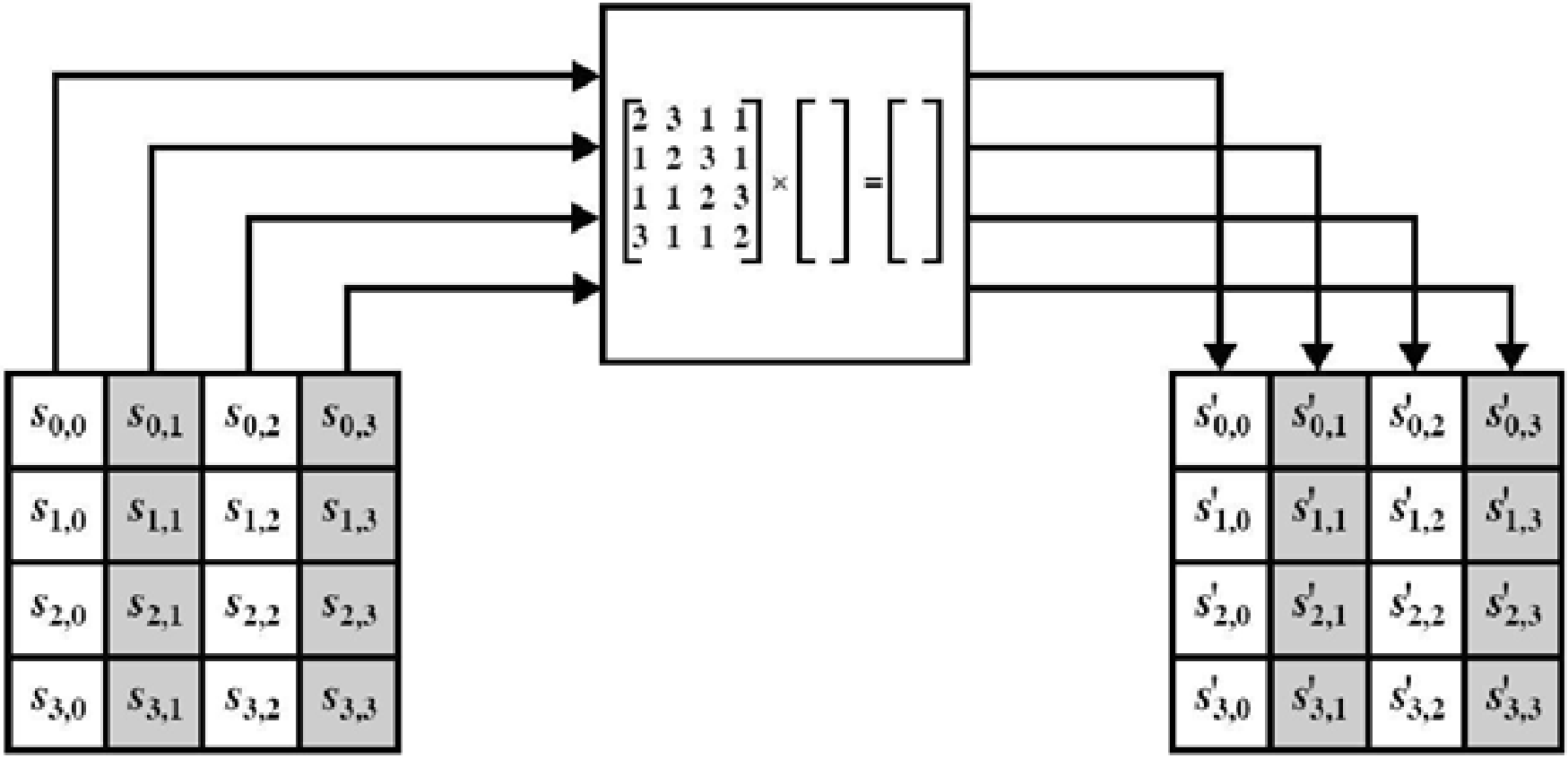


ShiftRows Transformation

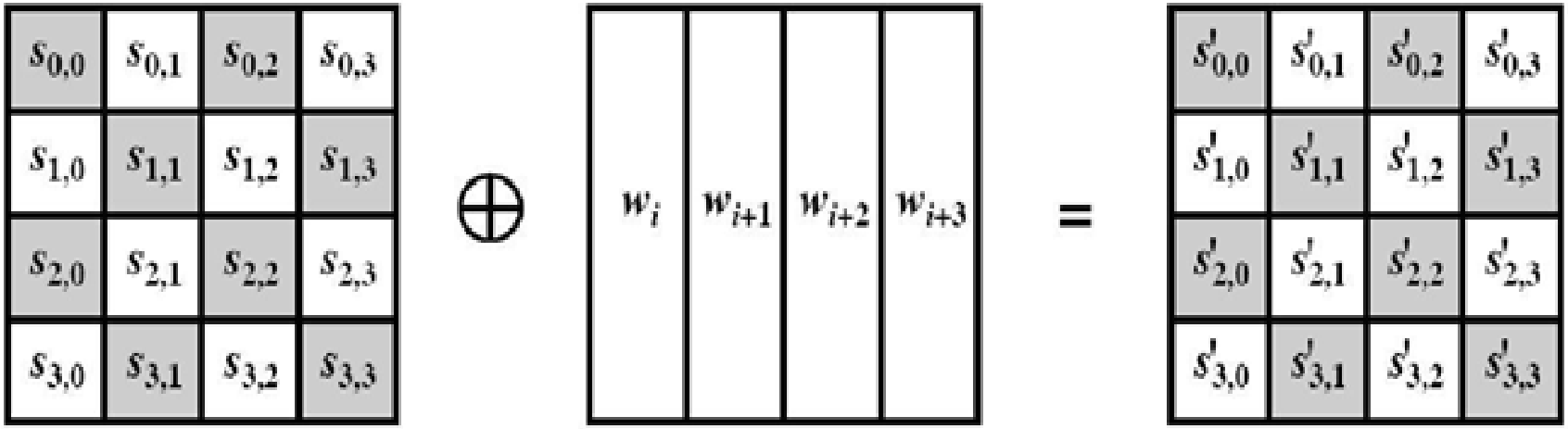


ShiftRows() cyclically shifts the last three rows in the State.

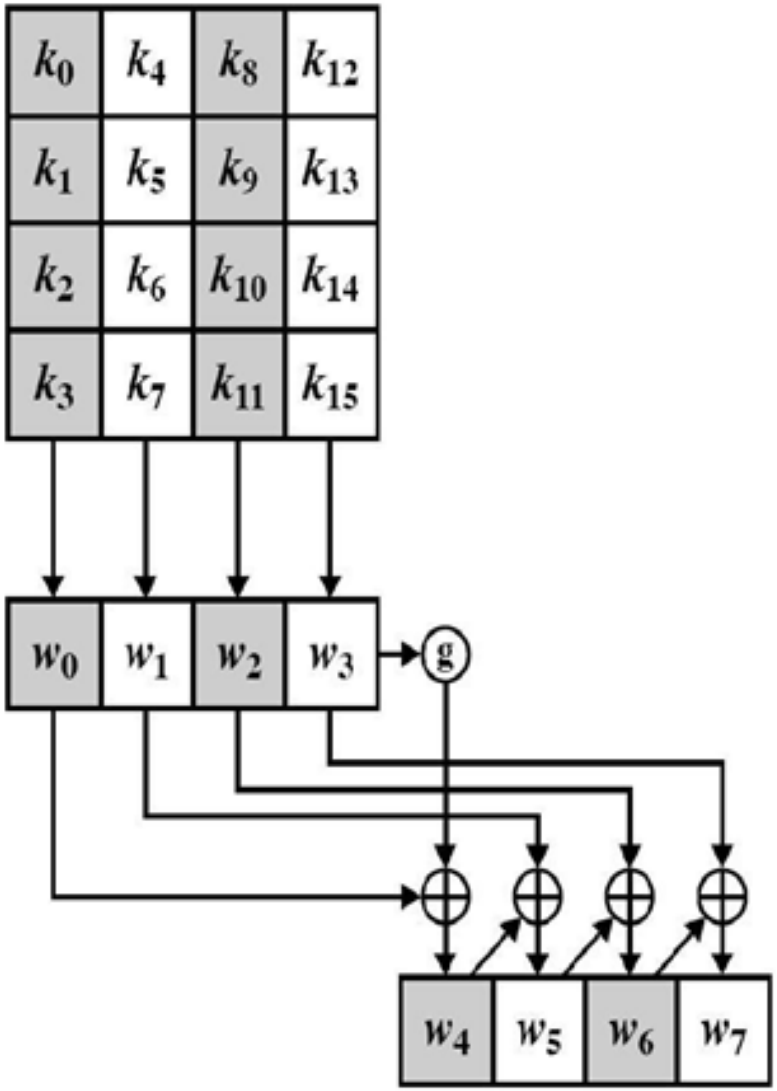
MixColumns Transformation



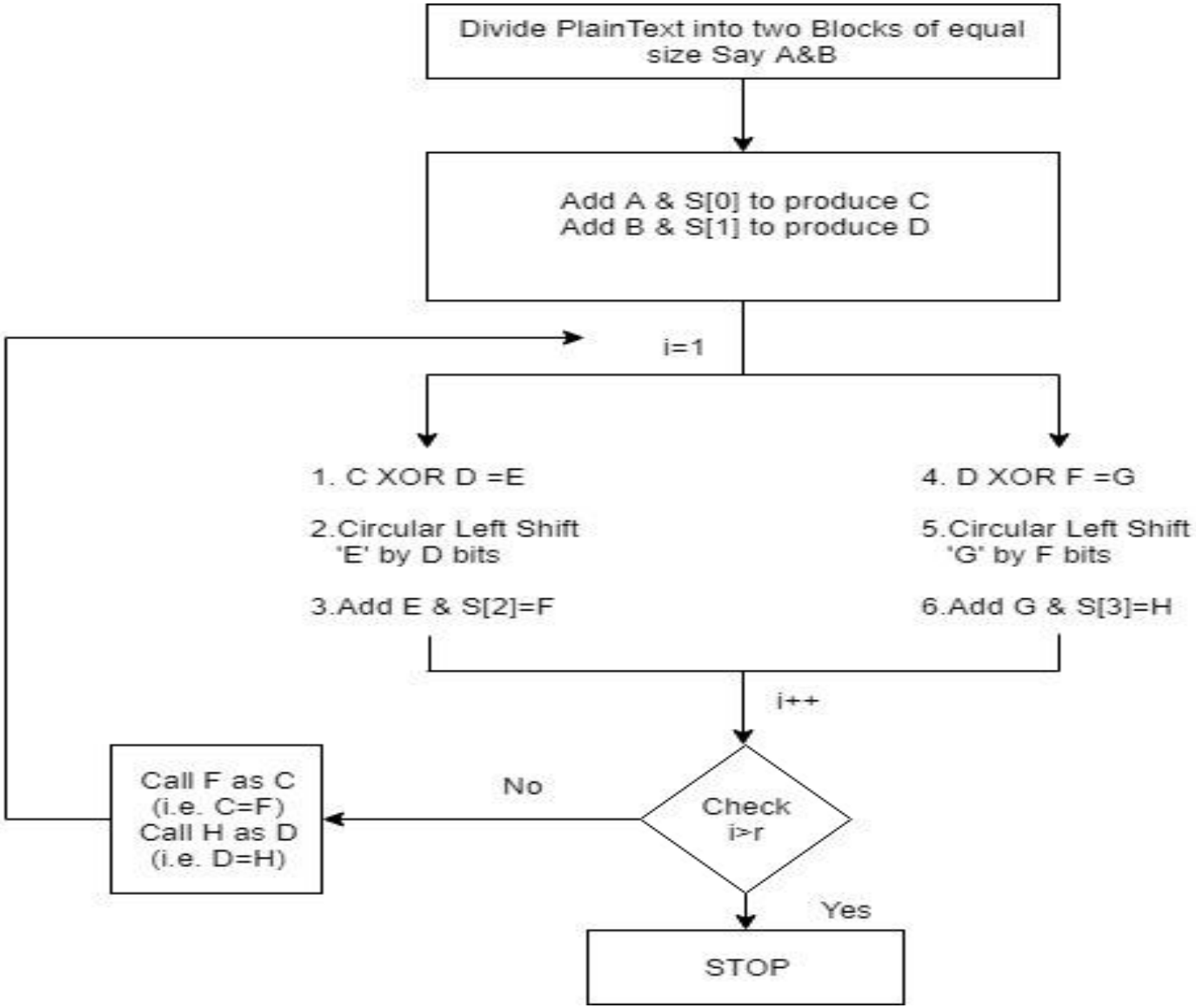
AddRoundKey Transformation



AES Key Expansion



RC-5 Algorithm



We divide the plaintext block into two equal parts A and B

Then they are XOR with two subkeys S{0} and S{1}

$$C = A + S[0]$$

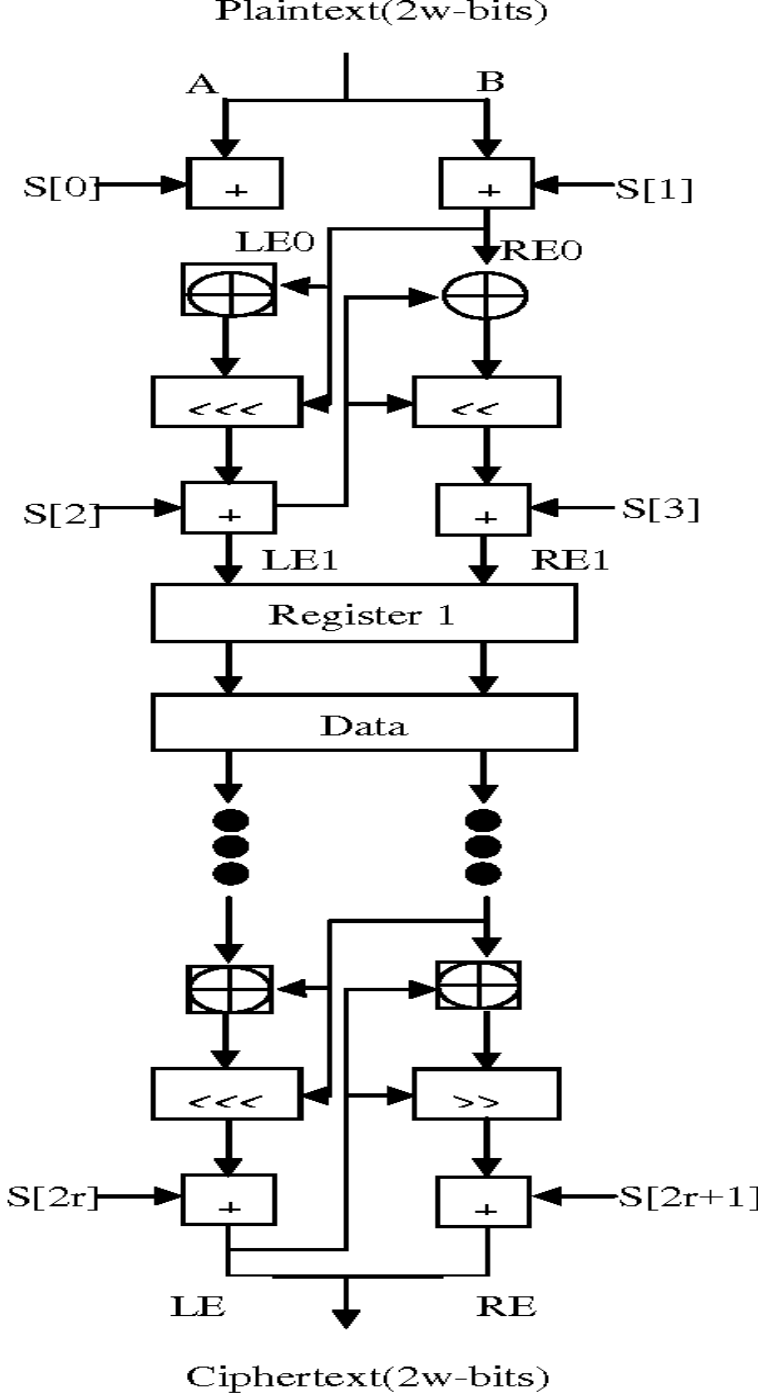
$$D = B + S[1]$$

for i = 1 to r do:

1. C $\oplus$ D = E
2. perform circular left shift on E by D bits
3. add E and S[2 * i] and store the result in F which is input for step 4
4. D $\oplus$ F = G
5. perform circular left shift on G by F bits
6. add G and S[2 * i + 1] and store the result in H
7. If i< r

Call F as C and H as D and repeat the steps from 1 to 7

else stop


Plaintext(2w-bits)
A
B
S[0]
+
+
S[1]
LE0
RE0
<<<
<<
S[2]
+
+
S[3]
LE1
RE1
Register 1
Data
<<<
>>
S[2r]
+
+
S[2r+1]
LE
RE
Ciphertext(2w-bits)

RC4 Algorithm

Key Scheduling:

```
 j = 0;
for i = 0 to 255 do
   {
        j = (j + S[i] + T[i])mod 256;
        Swap(S[i], S[j]);
    }
```

Stream Generation:

- i, j = 0;
- while (true)
- i = (i + 1)mod 256;
- j = (j + S[i])mod 256;
- Swap(S[i], S[j]);
- t = (S[i] + S[j])mod 256;
- k = S[t];

RSA ALGORITHM

- Choose two large prime numbers (p and q)
- Calculate n = p*q and Φn = (p-1)(q-1)
- Choose a number e where 1 < e < Φn
- Calculate d = e-1mod Φn
- You can bundle private key pair as (n,d)
- You can bundle public key pair as (n,e)

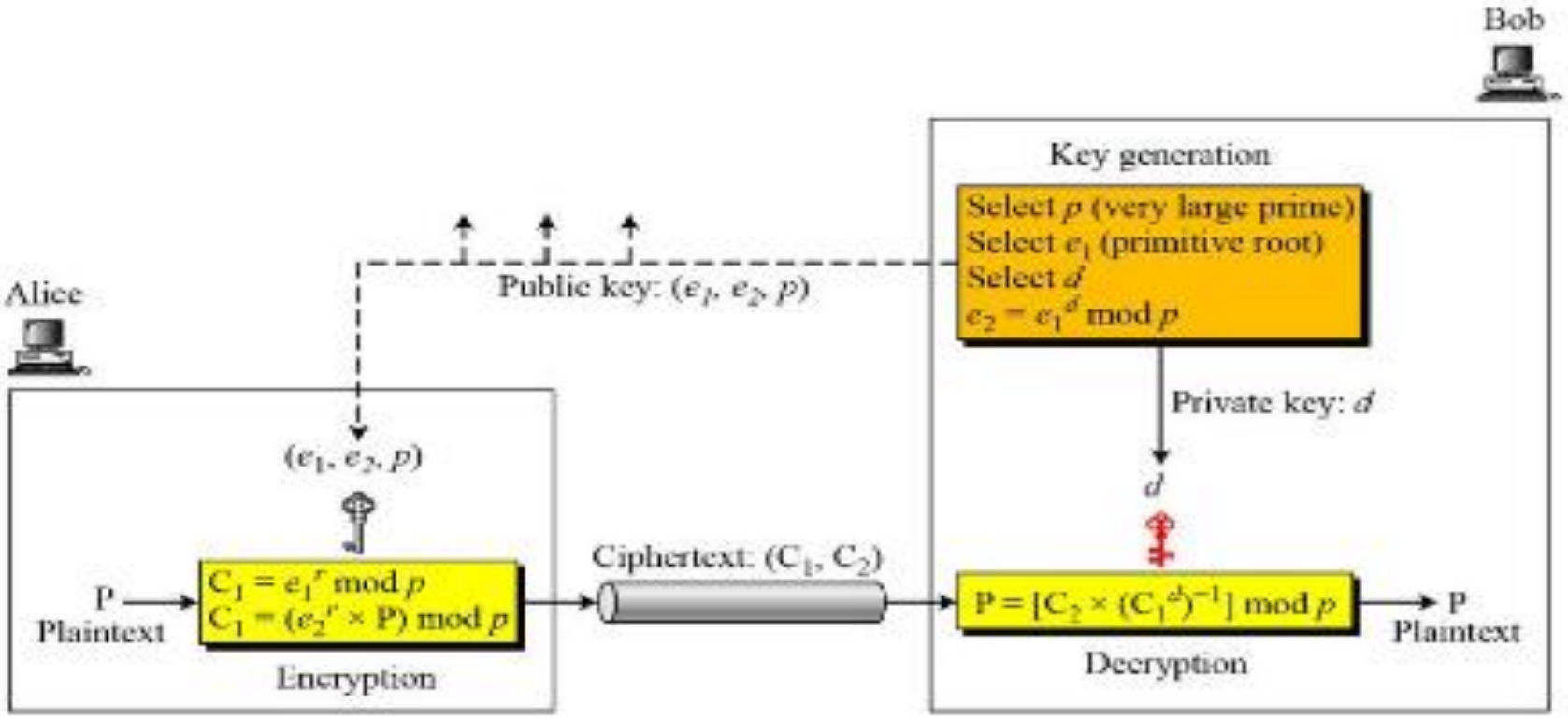
Encryption/Decryption Function

- If the plaintext is m, ciphertext = m^e mod n.
- If the ciphertext is c, plaintext = c^d mod n

Elgamal cryptography

El gamal cryptography works in 3 steps/stages

- Key generation
- Elgamal encryption
- Elgamal decryption



EIgamal cryptography Algorithm

A. EIgamal key generation:

- Select a large prime number ‘p’
- Select encryption key ‘e’1‘ to be primitive root of mod p
- Select decryption key ‘d’ such that 1≤d≤p−2
- Calcilate encryption key ‘e’2‘ such that
- e2=e1^d mod p
- Form the set i,e public key (e1,e2,p) to be announced publicly
- Private Key 'd' to be kept secret.

B. EIgamal key encryption:

- Select a random number ‘r’
- Compute the first part of ciphertext ‘c’1,c1=e1^rmodp‘
- Compute the second part of ciphertext ‘c’2,c2=(e2^r∗PT)modp

C. EIgamal key decryption:

- Calculate the PT
- PT=(c2∗(c1^d)$^{-1}$)modp

DIFFIE-HELLMAN KEY EXCHANGE

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Generation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

KNAPSACK ALGORITHM

Example –

- {1, 2, 4, 10, 20, 40} is a super increasing as
- 1<2, 1+2<4, 1+2+4<10, 1+2+4+10<20 and 1+2+4+10+20<40.

Derive the Public key

- **Step-1:**
  Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.

- **Step-2:**
  Choose two numbers n and m. Multiply all the values of private key by the number n and then find modulo m. The value of m must be greater than the sum of all values in private key, for example 110. And the number n should have no common factor with m, for example 31.

- **Step-3:**
  Calculate the values of Public key using m and n.
- 1x31 mod(110) = 31
- 2x31 mod(110) = 62
- 4x31 mod(110) = 14
- 10x31 mod(110) = 90
- 20x31 mod(110) = 70
- 40x31 mod(110) = 30
- Thus, our public key is {31, 62, 14, 90, 70, 30}
  And Private key is {1, 2, 4, 10, 20, 40}.

Now take an example for understanding the process of encryption and decryption.

Example –

Lets our plain text is 1001001111100101110.

1. Encryption :

As our knapsacks contain six values, so we will split our plain text in a groups of six:

- 100100 111100 101110

Multiply each values of public key with the corresponding values of each group and take their sum.

- 100100 {31, 62, 14, 90, 70, 30}
- 1x31+0x62+0x14+1x90+0x70+0x30 = 121

- 111100 {31, 62, 14, 90, 70, 30}
- 1x31+1x62+1x14+1x90+0x70+0x30 = 197

- 101110 {31, 62, 14, 90, 70, 30}
- 1x31+0x62+1x14+1x90+1x70+0x30 = 205
- So, our cipher text is 121 197 205.

2. Decryption :

The receiver receive the cipher text which has to be decrypt. The receiver also knows the values of m and n.

So, first we need to find the , which is multiplicative inverse of n mod m i.e.,

- n x mod(m) = 1
- 31 x mod(110) = 1 = >71

Now, we have to multiply 71 with each block of cipher text take modulo m.

- 121 x 71 mod(110) = 11

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e.,

1+10=11

so make that corresponding bits 1 and others 0 which is 100100.

Similarly,

- 197 x 71 mod(110) = 17
- 1+2+4+10=17 = 111100

- And, 205 x 71 mod(110) = 35
- 1+4+10+20=35 = 101110
- After combining them we get the decoded text. 100100111100101110 which is our plain text.