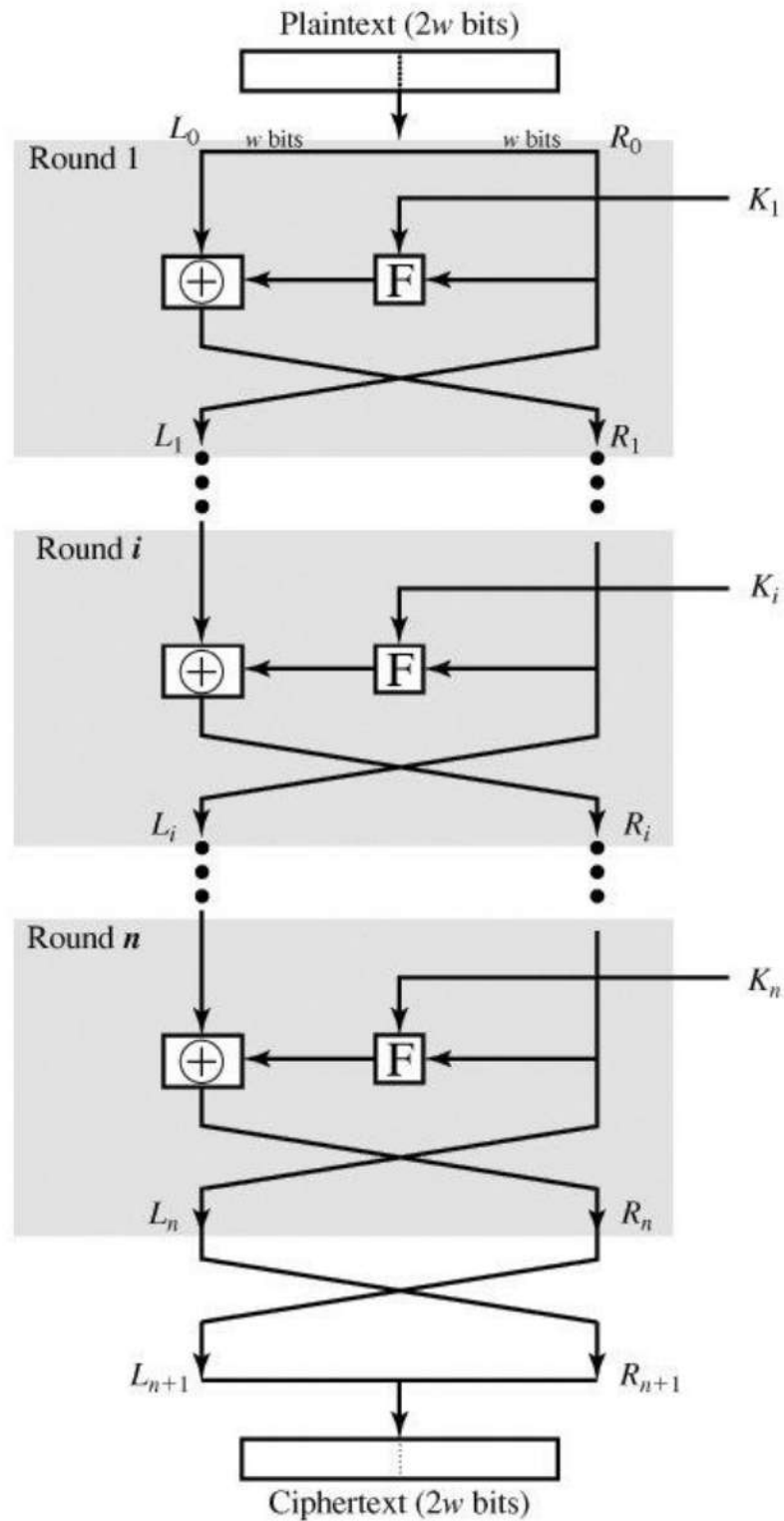


## Unit-II Symmetric Ciphers

### Fiestel Cipher Structure

Fiestel structure is shown in the following figure:



- Feistel cipher is a type of block cipher design, not a specific cipher.
- In a feistel cipher, the plaintext block  $P$  is divided into left and right halves:  $P = (L_0, R_0)$ . Then the two halves pass through  $n$  rounds of processing then combine to produce the cipher block.

For each round  $i = 1, 2, \dots, n$  new left and right halves are computed according to the rules:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus F(R_{i-1}, K_i) \end{aligned}$$

Where,  $F$  is round function and  $K_i$  is the subkey for round  $i$ .

The subkey is derived from the original key  $K_i$  according to a specified key schedule algorithm. Finally the ciphertext  $C$  is the output of the final round.

$$C = (L_n, R_n)$$

- All rounds have the same structure.
- 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 data followed by the XOR of the output of that function and left half of data.
- The **permutation** steps at each round swaps the modified  $L$  and unmodified  $R$ .
- The combination of substitution and permutation is called a round.

### Decryption

- The process of decryption is similar to encryption with reversed keys.

Start with ciphertext  $C = (L_n, R_n)$

For each round  $i = n, n - 1, \dots, 1$  compute:

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus F(R_{i-1}, K_i) \end{aligned}$$

Where,  $F$  is round function and  $K_i$  is subkey.

Plaintext:  $P = (L_0, R_0)$

### Design Features/Principles of Feistel Network:

- **Block size:** increasing size improves security, but reduced encryption/decryption speed.
- **Key size:** increasing key size improves security, makes exhaustive key searching harder, but may slow cipher.
- **Number of rounds:** increasing number of rounds increases security, but slows cipher.
- **Sub-key generation algorithm:** greater complexity in this algorithm can make analysis harder, but slows cipher.
- **Round function:** greater complexity can make analysis harder, but slows cipher.
- **Fast software en/decryption & ease of analysis:** are more recent concerns for practical use and testing.

### Substitution Permutation Network (SPN)

Substitution Permutation network is a series of linked mathematical operations used in block cipher algorithms. These networks describe a series of substitution and permutation operations to be applied on the block of input bits to produce output bits. It takes the block of the plaintext and key as inputs and applies several alternating “rounds” or “layers” of substitution boxes (S-boxes) and permutation boxes (P-boxes) to produce the ciphertext block.

- An **S-Box** substitutes a small block of bits (the input of the S-box) by another block of bits (the output of the S-box). In particular, the length of the output should be same as the length of the input. A good S-box will have the property that changing one input bit will change about half of the output bits. It will also have the property that each output bit will depend on every input bit.
- A **P-Box** is a permutation of all the bits: it takes the outputs of all the S-boxes of one round, permutes the bits, and feeds them into the S-boxes of the next round. A good P-box has the property that the output bits of any S-box are distributed to as many S-box inputs as possible.

In addition, at each round the key is combined using some group operation, typically XOR.

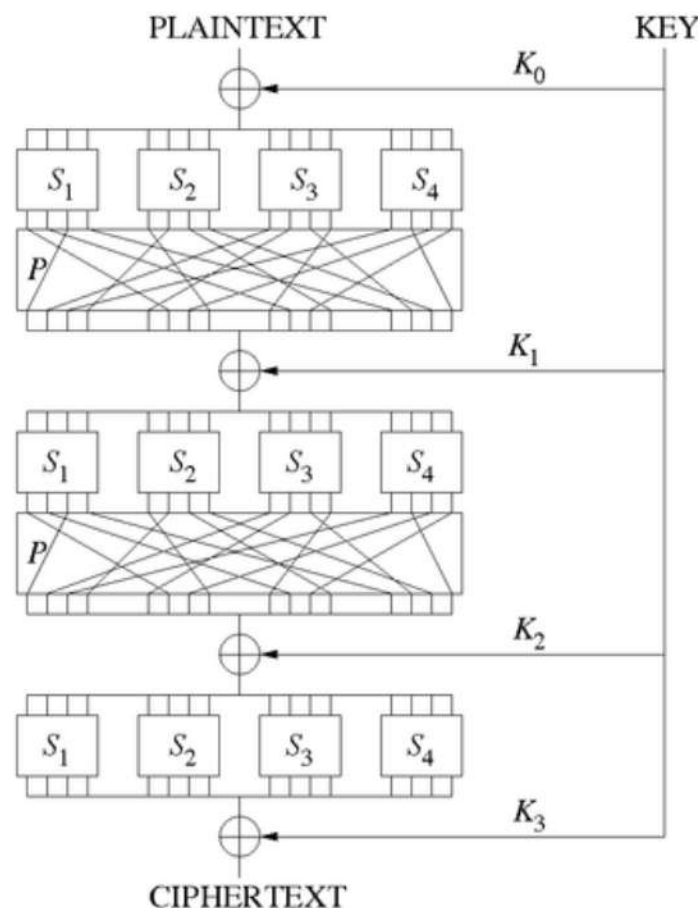


Fig: SPN with 3 rounds

- Provides confusion and diffusion of message and key.

### Decryption

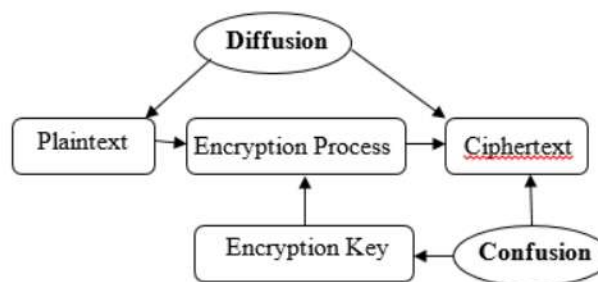
- Decryption is done by simply reversing the process (using the inverse of the S-boxes and P-boxes and applying the round keys in reverse order).



**Q. Explain the concepts of diffusion and confusion.****Solution:**

Block ciphers use “confusion” and “diffusion” in their encryption method. The terms diffusion and confusion were introduced by the famous information theorist Claude Shannon. According to the Shannon, there are two primitive operations with which strong encryption algorithms can be built:

- **Confusion** refers to making the relationship between the key and ciphertext as complex and involved as possible.
  - Each bit of the ciphertext should depend on several part of the key, obscuring the connections between the two.
  - E.g. S-box or substitution cipher.
- **Diffusion** means any of the characters in the plaintext is changed, then simultaneously several characters of the ciphertext should also be changed. Similarly, if the characters of ciphertext is changed then simultaneously several characters of plaintext should be changed.
  - Diffusion hides the relation between the ciphertext and the plaintext.
  - E.g. P-box or transposition cipher.

**Data Encryption Standards (DES)**

- DES is block cipher that operates on a plaintext block of 64 bits and returns ciphertext of same size.
- The key length is 56 bits. The key originally consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used for checking parity and are thereafter discarded.
- It consists of **16 rounds**, each round performs the steps of substitution (confusion) and transposition (diffusion).

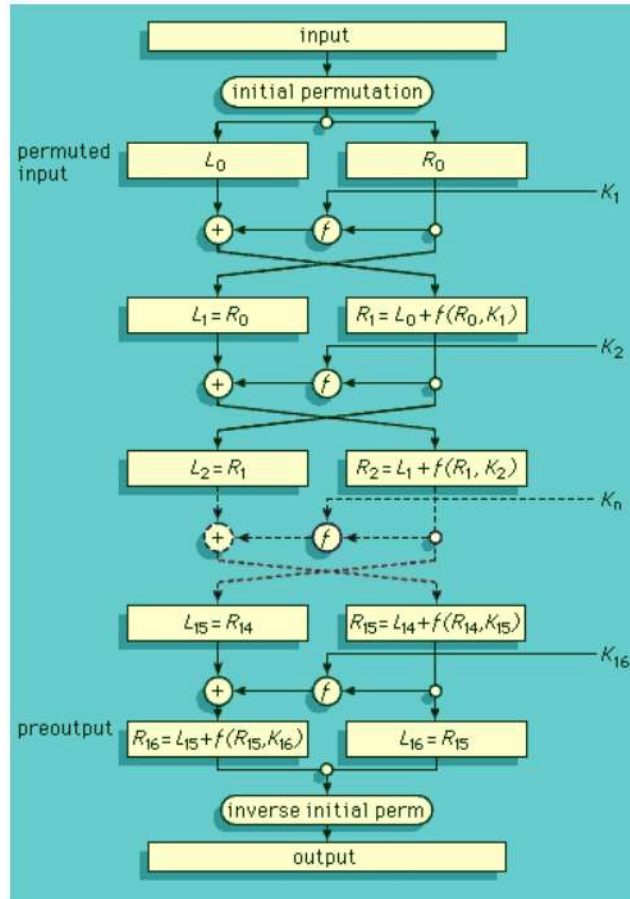
**DES Algorithm**

- The algorithm accepts plaintext  $P$ , performs an initial permutation,  $IP$ , on  $P$  producing  $P_0$ . The block is then broken down into left and right halves, the left ( $L_0$ ) being the first 32 bits of  $P_0$  and the right ( $R_0$ ) being the last 32 bits of  $P_0$ .
- With  $L_0$  and  $R_0$ , 16 rounds are performed until  $L_{16}$  and  $R_{16}$  are generated according the rule:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

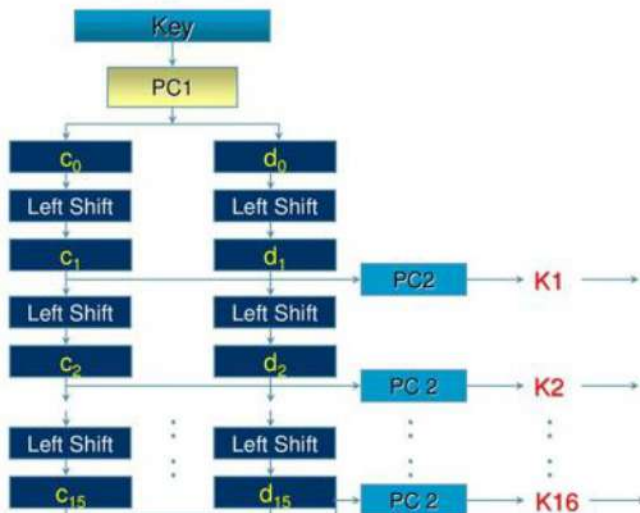
- The inverse permutation,  $IP^{-1}$ , is applied to  $L_{16}R_{16}$  to produce ciphertext  $C$ .



**Key Generation**

In DES encryption, the round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.

Initially, 56 bits of the key are selected from the initial 64-bit by Permuted Choice 1 (PC-1), the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves ( $C_0$  &  $D_0$ ); each half is thereafter treated separately. In successive rounds, both halves ( $C_{i-1}$  &  $D_{i-1}$ ) are rotated left by one or two bits specified for each round, and then 48 subkey bits are selected by Permuted Choice 2, PC-2 (24 bits from the left half, and 24 from the right) that serves as input to the function  $F(R_{i-1}, K_i)$ .

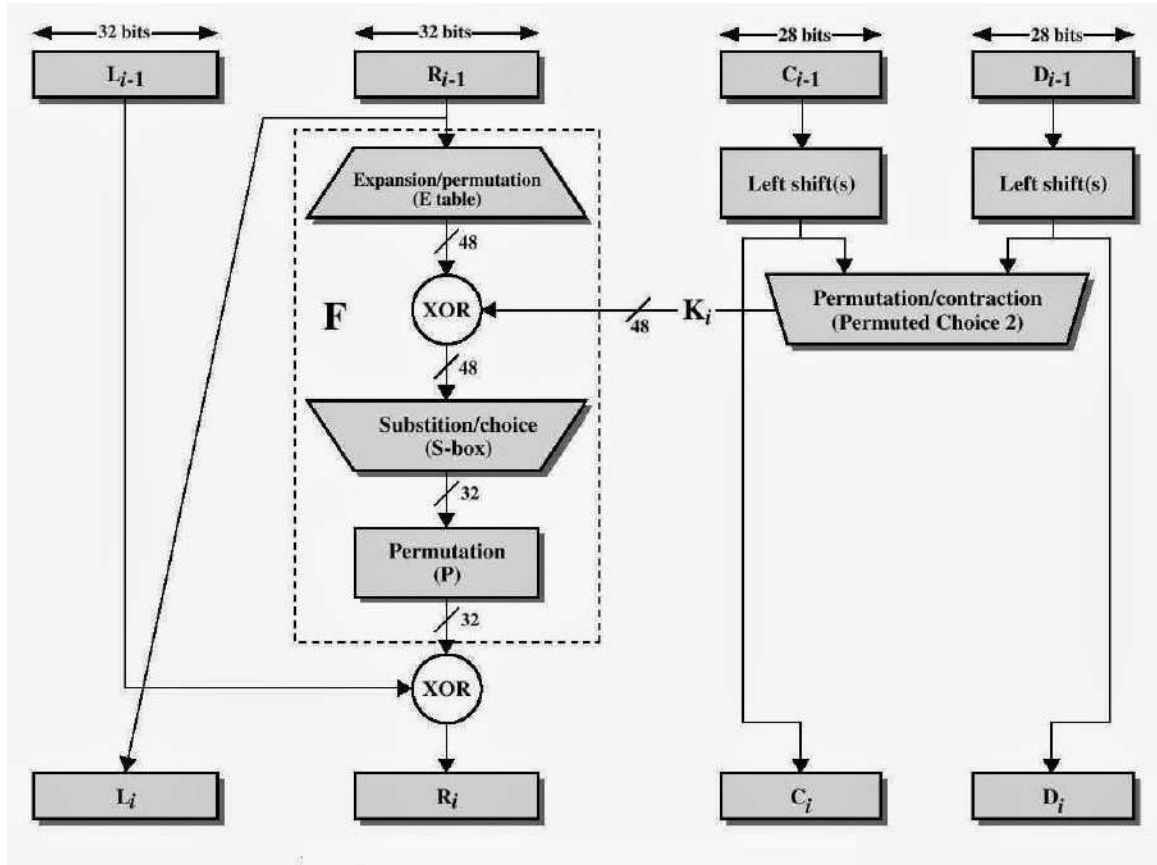


**Shifting**

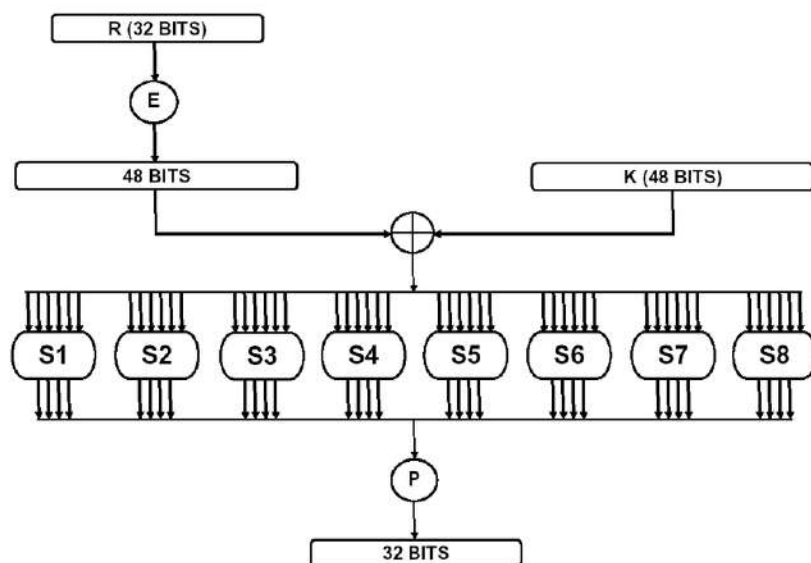
| Rounds      | Shift   |
|-------------|---------|
| 1, 2, 9, 16 | One bit |
| Others      | Two bit |

**Single Round of DES Algorithm**

The following figure shows a closer view of algorithms for a single iteration. The 64bit permuted input passes through 16 iterations, producing an intermediate 64-bit value at the conclusion of each iteration.



The left hand output of an iteration ( $L_i$ ) is equal to the right hand input to that iteration  $R_{i-1}$ . The right hand output  $R_i$  is exclusive OR of  $L_{i-1}$  and a complex function  $F$  of  $K_i$  and  $R_{i-1}$ . The function  $F$  can be depicted by the following figure.  $S_1, S_2, \dots, S_8$  represent the "S-boxes", which maps each combination of 48 input bits into a particular 32 bit pattern.





**Decryption of DES**

The process is the same as encryption, only the difference is that keys are used in reverse order. Thus, when in decryption mode, the key schedule algorithm has to generate the round keys as the sequence  $K_{16}, K_{15}, \dots, K_1$ .

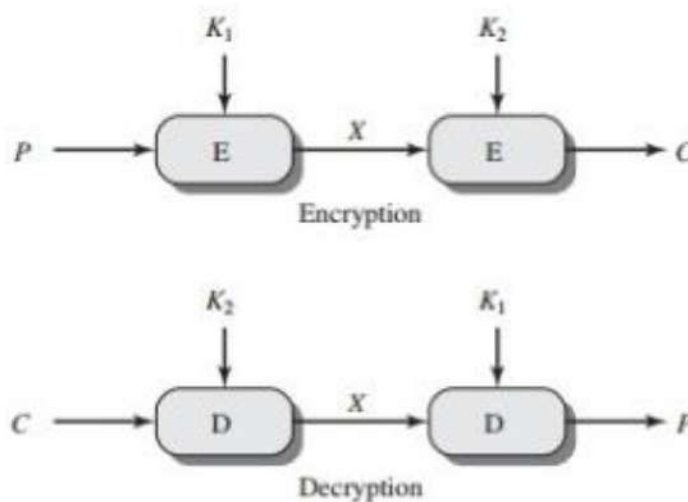
**Double DES**

- It does twice what DES normally does only once.
- It uses two keys  $K_1$  &  $K_2$  and encrypt the text using the two keys.

$$C = E_{K_2}(E_{K_1}(P))$$

- To decrypt simply use DES decryption twice.

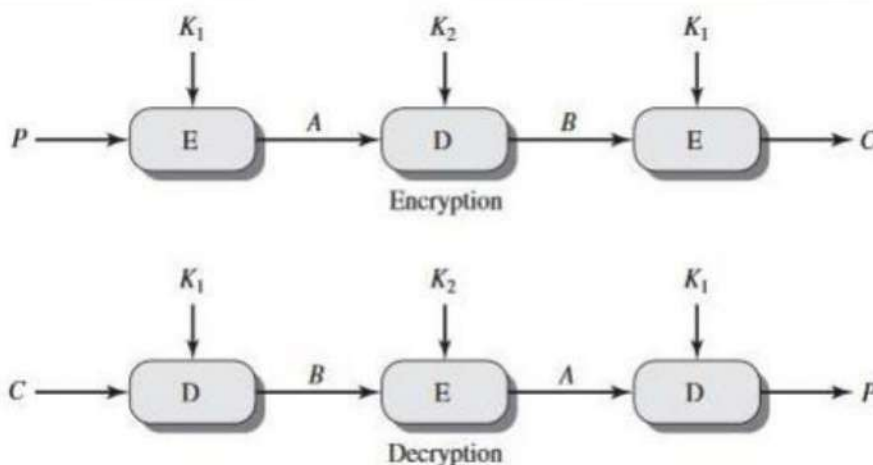
$$P = D_{K_1}(D_{K_2}(C))$$

**Triple DES**

- **With Two Keys:** It uses three stages of DES for encryption and decryption. The 1<sup>st</sup> & 3<sup>rd</sup> stages use  $K_1$  key and 2<sup>nd</sup> stage uses  $K_2$  key. To make triple DES compatible with single DES, the middle stage uses decryption in the encryption side and encryption in the decryption side.

$$\text{Encryption: } C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

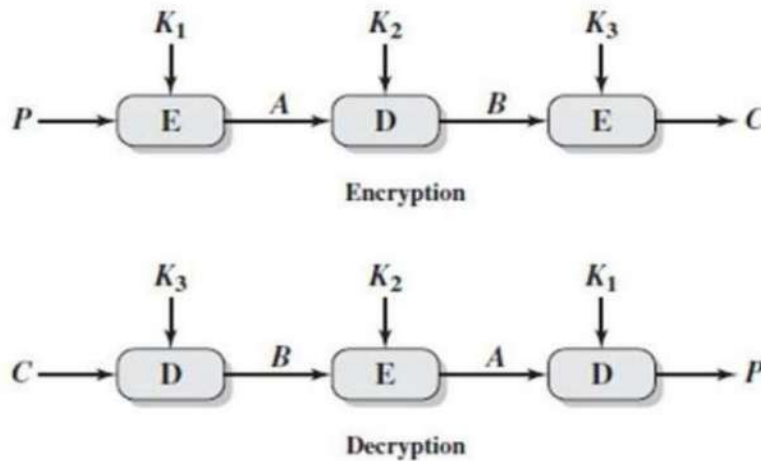
$$\text{Decryption: } P = D_{K_1}(E_{K_2}(D_{K_1}(C)))$$



- **With Three Keys:** It uses three stages of DES for encryption and decryption with three different keys.

$$\text{Encryption: } C = E_{K_3}(D_{K_2}(E_{K_1}(P)))$$

$$\text{Decryption: } P = D_{K_1}(E_{K_2}(D_{K_3}(C)))$$



#### Q. Which types of keys are considered as weak keys in DES?

##### Solution:

Weak keys are keys that cause the encryption mode of DES to act identically to the decryption mode of DES.

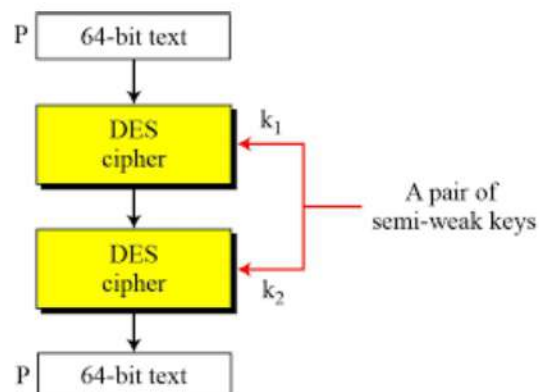
DES uses 16 48-bit subkeys generated from a master 56-bit key, one subkey is used in each of the sixteen DES rounds. The *weak keys* of DES are those which produce sixteen identical subkeys. This occurs when the key bits are:

- all zeros
- all ones
- the first half of the entire key is all ones and the second half is all zeros
- vice versa

Since all the subkeys are identical, encrypting twice produces the original plaintext.

#### Semi-weak Key

A semi-weak key creates only two different round keys each of them repeated eight times. There are six semi-weak key pairs. Specific pairs of keys have identical decryption. That is, there are two different keys,  $k_1$  and  $k_2$ , for which  $c = \text{DES}(p, k_1)$  and  $c = \text{DES}(p, k_2)$ . This similarity implies that  $k_1$  can decrypt a message encrypted under  $k_2$ .





## Finite Fields

### Groups

- denoted by  $\{G, \cdot\}$ , where  $\cdot$  is generic symbol and can be a binary symbol.
- is a set of elements with a binary operation, such that following axioms are obeyed:
  - $A_1$ : Closure  $\rightarrow$  if  $a, b \in G$ , then  $a \cdot b \in G$
  - $A_2$ : Associative  $\rightarrow a \cdot (b \cdot c) = (a \cdot b) \cdot c$  for all  $a, b, c \in G$
  - $A_3$ : Identity  $\rightarrow a \cdot e = e \cdot a$  for all  $a \in G$
  - $A_4$ : Inverse  $\rightarrow a \cdot a' = a' \cdot a = e$  for each  $a \in G$
  - $A_5$ : Commutative  $\rightarrow a \cdot b = b \cdot a$  for all  $a, b \in G$

$\rightarrow$  If  $A_1, A_2, A_3, A_4$  satisfies = Group

$\rightarrow$  If  $A_1, A_2, A_3, A_4, A_5$  satisfies = Abelian Group

### Rings

- denoted by  $\{R, +, *\}$
- is a set of elements with two binary operations addition and multiplication, such that following axioms are obeyed:
  - Abelian group under addition [ $A_1 - A_5$ ]
  - $M_1$ : Closure under multiplication  $\rightarrow$  if  $a, b \in R$ , then  $ab \in R$
  - $M_2$ : Associativity of multiplication  $\rightarrow a(bc) = (ab)c$  for all  $a, b, c \in R$
  - $M_3$ : Distributive  $\rightarrow a(b + c) = ab + ac$  for all  $a, b, c \in R$
  - $M_4$ : Commutative  $\rightarrow ab = ba$  for all  $a, b \in R$
  - $M_5$ : Multiplicative identity  $\rightarrow ae = ea = a$  for all  $a \in R$
  - $M_6$ : No zero divisor  $\rightarrow$  if  $ab = 0$  then either  $a = 0$  or  $b = 0, a, b \in R$

$\rightarrow$  If  $A_1 - M_4$  satisfies = Commutative ring

$\rightarrow$  If  $A_1 - M_6$  satisfies = Integral domain

### Fields

- denoted by  $\{F, +, *\}$
- is a set of elements with two binary operations addition and multiplication, such that following axioms are obeyed:
  - [ $A_1 - M_6$ ]
  - $M_7$ : Multiplicative inverse  $\rightarrow aa^{-1} = a^{-1}a = 1$

### Modular Arithmetic

- According to division rule,

$$\begin{array}{r} p) n (q \\ \underline{-pq} \\ r \end{array} \quad \boxed{n = qp + r}$$

Where,  $p$  is divisor,  $q$  is quotient and  $r$  is remainder.

- **Mod** is an operator that gives the remainder.

**E.g.**

$$\begin{array}{lll} 5 \bmod 2 = 1, & 7 \bmod 9 = 7, & -7 \bmod 9 = 2, \\ 15 \bmod 3 = 0, & -6 \bmod 7 = 1, & 81 \bmod 7 = 4, \\ -13 \bmod 10 = 7 & & \end{array}$$

### Congruent Modulo

Two integers  $a$  and  $b$  are said to be congruent modulo  $n$ , if  $a \equiv b \pmod{n}$  i.e. when  $a$  is divided by  $n$ , we get remainder  $b$ .

**E.g.**  $7 \equiv 2 \pmod{5}$

$$a \equiv b \pmod{n} \Leftrightarrow (a - b) \bmod n = 0 \Leftrightarrow (a \bmod n) = (b \bmod n)$$

### Greatest Common Divisor (GCD)

The GCD of two positive integers  $a$  and  $b$  is the greatest number that divides both  $a$  and  $b$ .

E.g.  $GCD(25, 10) = 5$

**Euclidean algorithm to find GCD:**

```

 $r_1 = a;$ 
 $r_2 = b;$ 
while( $r_2 > 0$ )
{
 $q = r_1/r_2;$ 
 $r = r_1 - q * r_2;$ 
 $r_2 = r;$ 
 $r_1 = r_2;$ 
}
 $GCD(a, b) = r_1$ 

```

**Q.  $GCD(161, 28) = ?$**

**Sol<sup>n</sup>:**

Here,

$$a = 161, b = 28$$

Now,

| $r_1$ | $r_2$ | $q$ | $r$ |
|-------|-------|-----|-----|
| 161   | 28    | 5   | 21  |
| 28    | 21    | 1   | 7   |
| 21    | 7     | 3   | 0   |
| 7     | 0     |     |     |

$$\therefore GCD(161, 28) = 7$$

**Q. GCD(60, 25) = ?**

**Sol<sup>n</sup>:**

Here,

$$a = 60, b = 25$$

Now,

| $r_1$ | $r_2$ | $q$ | $r$ |
|-------|-------|-----|-----|
| 60    | 25    | 2   | 10  |
| 25    | 10    | 2   | 5   |
| 10    | 5     | 2   | 0   |
| 5     | 0     |     |     |

$$\therefore \text{GCD}(60, 25) = 5$$

### Set of Residues

- denoted as  $Z_n$ .
- is a set of remainders when divided by  $n$  i.e.  $Z_n = \{0, 1, 2, 3, \dots, n-1\}$ .

**E.g.**  $Z_5 = \{0, 1, 2, 3, 4\}$

$$Z_2 = \{0, 1\}$$

### Operations of $Z_n$

- Addition, subtraction, multiplication

1. Add 7 to 14 in  $Z_{15}$ .

$$(7 + 14) \text{ mod } 15 = 21 \text{ mod } 15 = 6$$

2. Subtract 11 from 7 in  $Z_{13}$ .

$$(7 - 11) \text{ mod } 13 = -4 \text{ mod } 13 = 9$$

3. Multiply 11 by 7 in  $Z_{20}$ .

$$(11 * 7) \text{ mod } 20 = 77 \text{ mod } 20 = 17$$

### Properties of Modular arithmetic for integers in $Z_n$

| Property                  | Expression   |
|---------------------------|--|
| Commutative Laws          | $(w + x) \text{ mod } n = (x + w) \text{ mod } n$<br>$(w \times x) \text{ mod } n = (x \times w) \text{ mod } n$                                   |
| Associative Laws          | $[(w + x) + y] \text{ mod } n = [w + (x + y)] \text{ mod } n$<br>$[(w \times x) \times y] \text{ mod } n = [w \times (x \times y)] \text{ mod } n$ |
| Distributive Law          | $[w \times (x + y)] \text{ mod } n = [(w \times x) + (w \times y)] \text{ mod } n$   |
| Identities                | $(0 + w) \text{ mod } n = w \text{ mod } n$<br>$(1 \times w) \text{ mod } n = w \text{ mod } n$  |
| Additive Inverse ( $-w$ ) | For each $w \in Z_n$ , there exists a $z$ such that $w + z = 0 \text{ mod } n$   |

**Residue Class**

- denoted as  $[a]$  or  $[a]_n$ .
- is the set of integers, when divided by  $n$ , we get remainder  $a$ .
- i.e.  $x \in [a]_n ; x \equiv a \pmod{n}$

**E.g.**Let  $n = 4$ 

$$[0] = \{ \dots \dots \dots, -12, -8, -4, 0, 4, 8, 12, \dots \dots \dots \}$$

$$[1] = \{ \dots \dots \dots, -11, -7, -3, 1, 5, 9, 13, \dots \dots \dots \}$$

$$[2] = \{ \dots \dots \dots, -10, -6, -2, 2, 6, 10, 14, \dots \dots \dots \}$$

$$[3] = \{ \dots \dots \dots, -9, -5, -1, 3, 7, 11, 15, \dots \dots \dots \}$$

**Quadratic Residue**

- Suppose ' $p$ ' is an odd prime and ' $a$ ' is an integer.
- ' $a$ ' is defined to be quadratic residue if  $y^2 \equiv a \pmod{p}$ , where  $y \in Z_p$

**E.g.**

$$p = 7$$

$$Z_7 = \{1, 2, 3, 4, 5, 6\}$$

$$1^2 \equiv 1 \pmod{7}$$

$$2^2 \equiv 4 \pmod{7}$$

$$3^2 \equiv 2 \pmod{7}$$

$$4^2 \equiv 2 \pmod{7}$$

$$5^2 \equiv 4 \pmod{7}$$

$$6^2 \equiv 1 \pmod{7}$$

$\therefore 1, 2, 4$  are quadratic residue modulo 7.

**Additive inverse of  $Z_n$** 

Let  $a, b \in Z_n$ , then  $a$  is additive inverse of  $b$  if  $(a + b) \pmod{n} = 0$ .

**E.g.**

**1. Find all the additive inverse pairs in  $Z_5$ .**

**Sol<sup>n</sup>:**

$$Z_5 = \{0, 1, 2, 3, 4\}$$

The additive inverse pairs in  $Z_5$  are:

$$(0, 0), \text{ since } (0+0) \pmod{5} = 0.$$

$$(1, 4), \text{ since } (1+4) \pmod{5} = 0.$$

$$(2, 3), \text{ since } (2+3) \pmod{5} = 0.$$

$$(3, 2), \text{ since } (3+2) \pmod{5} = 0.$$

$$(4, 1), \text{ since } (4+1) \pmod{5} = 0.$$



**2. Find the additive inverse of all the elements of  $Z_{10}$ .****Sol<sup>n</sup>:**

$$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Now,

|           |   |   |   |   |   |   |   |   |   |   |
|-----------|---|---|---|---|---|---|---|---|---|---|
| <b>w</b>  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| <b>-w</b> | 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Where, -w is the additive inverse of w.

**Multiplicative inverse in  $Z_n$** Let  $a, b \in Z_n$ , then a is multiplicative inverse of b if  $(a * b) \bmod n = 1$ .**E.g.****1. Find all the multiplicative inverse pairs in  $Z_{10}$ .****Sol<sup>n</sup>:**

$$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

The multiplicative inverse pairs in  $Z_{10}$  are;(1, 1), since  $(1 * 1) \bmod 10 = 1$ .(3, 7), since  $(3 * 7) \bmod 10 = 1$ .(7, 3), since  $(7 * 3) \bmod 10 = 1$ .(9, 9), since  $(9 * 9) \bmod 10 = 1$ .**2. Find multiplicative inverse of each nonzero element in  $Z_7$ .****Sol<sup>n</sup>:**

$$Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$$

Now,

|                       |   |   |   |   |   |   |
|-----------------------|---|---|---|---|---|---|
| <b>w</b>              | 1 | 2 | 3 | 4 | 5 | 6 |
| <b>w<sup>-1</sup></b> | 1 | 4 | 5 | 2 | 3 | 6 |

Where,  $w^{-1}$  is the multiplicative inverse of w.**Co-prime**

- Also called **relatively prime**.
- Two positive numbers a and b are co-prime of each other if and only if  $GCD(a, b) = 1$ .

**E.g.** (10, 3), (22, 13)**Note:** Let  $b \in Z_n$  then the multiplicative inverse of b exist if and only if  $GCD(b, n) = 1$  i.e. b and n are co-prime.**Q. Find the multiplicative inverse of 2 in  $Z_6$ .****Sol<sup>n</sup>:**Since 2 and 6 are not co-prime so multiplicative inverse of 2 in  $Z_6$  doesn't exist.

**Extended Euclidean Algorithm (EEA) to find multiplicative inverse**

To find the multiplicative inverse of  $b$  in  $Z_n$ , EEA works as follows:

```

 $r_1 = n;$ 
 $r_2 = b;$ 
 $t_1 = 0;$ 
 $t_2 = 1;$ 
while( $r_2 > 0$ )
{
 $q = r_1/r_2;$ 
 $r = r_1 - q * r_2;$ 
 $r_1 = r_2;$ 
 $r_2 = r;$ 
 $t = t_1 - q * t_2;$ 
 $t_1 = t_2;$ 
 $t_2 = t;$ 
}
if( $r_1 == 1$ ) then  $b^{-1} = t_1 \text{ mod } n.$ 

```

**E.g.**

**Q. Find the multiplicative inverse of 11 in  $Z_{26}$ .**

**Sol<sup>n</sup>:**

Here,  $b = 11$ ,  $Z_n = Z_{26}$  i.e.  $n = 26$

Now,

| $q$ | $r_1$ | $r_2$ | $r$ | $t_1$ | $t_2$ | $t$ |
|-----|-------|-------|-----|-------|-------|-----|
| 2   | 26    | 11    | 4   | 0     | 1     | -2  |
| 2   | 11    | 4     | 3   | 1     | -2    | 5   |
| 1   | 4     | 3     | 1   | -2    | 5     | -7  |
| 3   | 3     | 1     | 0   | 5     | -7    | 26  |
|     | 1     | 0     |     | -7    | 26    |     |

$$\therefore 11^{-1} = -7 \text{ mod } 26 = 19$$

**Q. Find the multiplicative inverse of 23 in  $Z_{100}$ .**

**Sol<sup>n</sup>:**

Here,  $b = 23$ ,  $Z_n = Z_{100}$  i.e.  $n = 100$

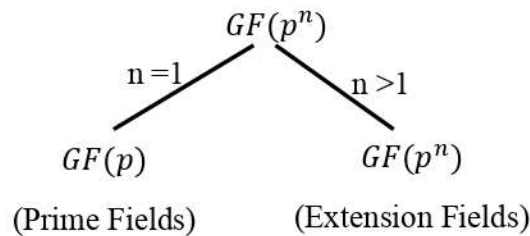
Now,

| $q$ | $r_1$ | $r_2$ | $r$ | $t_1$ | $t_2$ | $t$ |
|-----|-------|-------|-----|-------|-------|-----|
| 4   | 100   | 23    | 8   | 0     | 1     | -4  |
| 2   | 23    | 8     | 7   | 1     | -4    | 9   |
| 1   | 8     | 7     | 1   | -4    | 9     | -13 |
| 7   | 7     | 1     | 0   | 9     | -13   | 100 |
|     | 1     | 0     |     | -13   | 100   |     |

$$\therefore 23^{-1} = -13 \text{ mod } 100 = 87$$

**Galois Field (GF)**

- A Galois field can be defined as a set of numbers that we can add, subtract, multiply and divide together and only ever end up with a result that exists in our set of numbers.
- The number of elements of a Galois field is of the form  $p^n$ , where  $p$  is a prime and  $n$  is a positive integer. Generally, it is denoted by  $GF(p^n)$ .



For a given prime  $p$ ,  $GF(p)$  is defined as the set  $Z_p = \{0, 1, 2, \dots, p - 1\}$  of integers together with arithmetic operations modulo  $p$ .

*E.g.* Arithmetic in  $GF(7)$

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 0 |
| 2 | 2 | 3 | 4 | 5 | 6 | 0 | 1 |
| 3 | 3 | 4 | 5 | 6 | 0 | 1 | 2 |
| 4 | 4 | 5 | 6 | 0 | 1 | 2 | 3 |
| 5 | 5 | 6 | 0 | 1 | 2 | 3 | 4 |
| 6 | 6 | 0 | 1 | 2 | 3 | 4 | 5 |

(a) Addition modulo 7

|   | w                       | -w | w <sup>-1</sup> |
|---|-------------------------|----|-----------------|
| 0 | 0                       | 0  | —               |
| 1 | 1                       | 6  | 1               |
| 2 | 2                       | 5  | 4               |
| 3 | 3                       | 4  | 5               |
| 4 | 4 <td>3</td> <td>2</td> | 3  | 2               |
| 5 | 5                       | 2  | 3               |
| 6 | 6                       | 1  | 6               |

(c) Additive and multiplicative inverses modulo 7

| × | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 0 | 2 | 4 | 6 | 1 | 3 | 5 |
| 3 | 0 | 3 | 6 | 2 | 5 | 1 | 4 |
| 4 | 0 | 4 | 1 | 5 | 2 | 6 | 3 |
| 5 | 0 | 5 | 3 | 1 | 6 | 4 | 2 |
| 6 | 0 | 6 | 5 | 4 | 3 | 2 | 1 |

(b) Multiplication modulo 7

**Polynomial Arithmetic**

In general, polynomial is an expression of the form

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 + a_0$$

for some non-negative integer  $n$  and where the coefficient  $a_0, a_1, \dots, a_n$  are drawn from some designated set  $S$ .  $S$  is called the coefficient set.

When  $a_n \neq 0$ , we have a polynomial of degree  $n$ .

Polynomial arithmetic deals with the addition, subtraction, multiplication and division of polynomials.

**Addition:**

$$\begin{aligned} f(x) &= a_2x^2 + a_1x + a_0 \\ g(x) &= b_1x + b_0 \\ f(x) + g(x) &= a_2x^2 + (a_1 + b_1)x + (a_0 + b_0) \end{aligned}$$

**Subtraction:**

$$\begin{aligned} f(x) &= a_2x^2 + a_1x + a_0 \\ g(x) &= b_3x^3 + b_0 \\ f(x) - g(x) &= -b_3x^3 + a_2x^2 + a_1x + (a_0 - b_0) \end{aligned}$$

**Multiplication:**

$$\begin{aligned} f(x) &= a_2x^2 + a_1x + a_0 \\ g(x) &= b_1x + b_0 \\ f(x) * g(x) &= a_2b_1x^3 + (a_2b_0 + a_1b_1)x^2 + (a_1b_0 + a_0b_1) \end{aligned}$$

**Division:**

$$\begin{aligned} f(x) &= a_2x^2 + a_1x + a_0 \\ g(x) &= b_1x + b_0 \\ \frac{f(x)}{g(x)} &=? \text{ (Obtained by long division)} \end{aligned}$$

**Long division for polynomials consists of the following steps:**

- Arrange both the dividend and the divisor in the descending powers of the variable.
- Divide the first term of the dividend by the first term of the divisor and write the result as the first term of the quotient.
- Multiply the divisor with the quotient term just obtained and arrange the result under the dividend so that the same powers of  $x$  match up. Subtract the expression just laid out from the dividend.
- Consider the result of the above subtraction as the new dividend and go back to the first step.

**Polynomial Arithmetic over  $GF(p)$**

Polynomial arithmetic in which the arithmetic on the coefficients is performed modulo  $p$ ; that is, the coefficients are in  $GF(p)$ .

E.g.

**Q. Calculate the result of the following if the polynomials are over  $GF(2)$ .**

- $(x^7 + x^5 + x^4 + x^3 + x + 1) + (x^3 + x + 1)$
- $(x^7 + x^5 + x^4 + x^3 + x + 1) - (x^3 + x + 1)$
- $(x^7 + x^5 + x^4 + x^3 + x + 1) \times (x^3 + x + 1)$
- $(x^7 + x^5 + x^4 + x^3 + x + 1)/(x^3 + x + 1)$

**Sol<sup>n</sup>:**





**International Data Encryption Standard (IDEA)**

The block cipher IDEA operates with 64-bit plaintext and ciphertext blocks and is controlled by a 128-bit key from which we derive 52 subkeys that is used in the algorithm, and consists of a series of eight identical transformations (a round) in which 6 different subkeys are used and last four keys are used for output transformation (the half-round).

IDEA derives much of its security by interleaving operations from different groups - modular addition and multiplication, and bitwise eXclusive OR (XOR) - which are algebraically "incompatible" in some sense. In more detail, these operators, which all deal with 16-bit quantities, are:

- ⊕ bitwise XOR of 16-bit sub-blocks
- ⊞ addition modulo  $2^{16}$  of 16-bit integers
- ⊙ multiplication modulo  $2^{16} + 1$  of 16-bit integers with the zero sub-block corresponding to  $2^{16}$

After the eight rounds comes a final "half round", for the output.

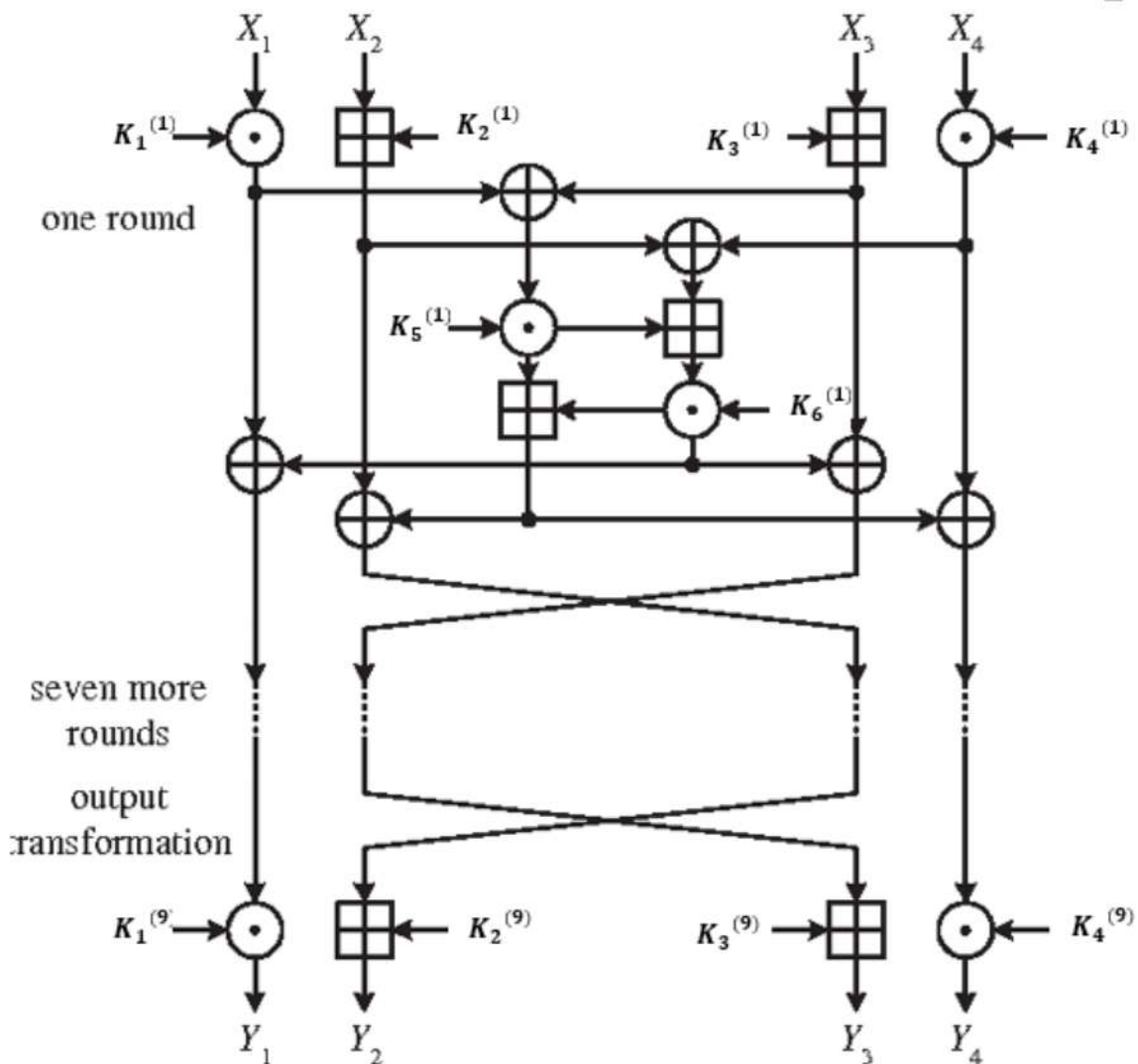


Fig: Structure of IDEA

### **Key Generation**

The 64-bit plaintext block is partitioned into four 16-bit sub-blocks, since all the algebraic operations used in the encryption process operate on 16-bit numbers. For each of the encryption round, six 16-bit key sub-blocks are generated from the 128-bit key. Since a further four 16-bit key-sub-blocks are required for the subsequent output transformation, a total of 52 (= 8 x 6 + 4) different 16-bit sub-blocks have to be generated from the 128-bit key.

The 52 16-bit key sub-blocks which are generated from the 128-bit key are produced as follows:

- First, the 128-bit key is partitioned into eight 16-bit sub-blocks which are then directly used as the first eight key sub-blocks.
- The 128-bit key is then cyclically shifted to the left by 25 positions, after which the resulting 128-bit block is again partitioned into eight 16-bit sub-blocks to be directly used as the next eight key sub-blocks.
- The cyclic shift procedure described above is repeated until all of the required 52 16-bit key sub-blocks have been generated.

### **Encryption**

The process consists of eight identical encryption steps (known as encryption rounds) followed by an output transformation.

- The first four 16-bit key sub-blocks are combined with two of the 16-bit plaintext blocks using addition modulo  $2^{16}$ , and with the other two plaintext blocks using multiplication modulo  $2^{16} + 1$ .
- At the end of the first encryption round four 16-bit values are produced which are used as input to the second encryption round.
- The process is repeated in each of the subsequent 7 encryption rounds.
- The four 16-bit values produced at the end of the 8<sup>th</sup> encryption round are combined with the last four of the 52 key sub-blocks using addition modulo  $2^{16}$  and multiplication modulo  $2^{16} + 1$  to form the resulting four 16-bit ciphertext blocks.

### **Decryption**

- The computational process used for decryption of the cipher text is essentially the same as that used for encryption of the plaintext.
- The only difference is that each of the 52 16-bit key sub-blocks used for decryption is the inverse of the key sub-block used during encryption in respect of the applied algebraic group operation.
- Additionally, the key sub-blocks must be used in the reverse order during decryption in order to reverse the encryption process.

**Q. Explain odd and even round (round operations) in IDEA.**

**Sol<sup>n</sup>:**

IDEA uses 8 full rounds and 1 halfround. We now break the 8 full round and make it 16 rounds such that there are total 17 rounds where 9 odd rounds (1, 3, ..., 17) are identical and 8 even rounds (2, 4, ..., 16) are identical. Each odd round takes 4 subkeys and each even round takes 2 subkeys.

**Odd Round**

- Treat 64-bit input block as four 16-bit sub-blocks ( $X_a, X_b, X_c, X_d$ ).
- Uses 4 16-bit keys ( $K_a, K_b, K_c, K_d$ ).

It works as follow:

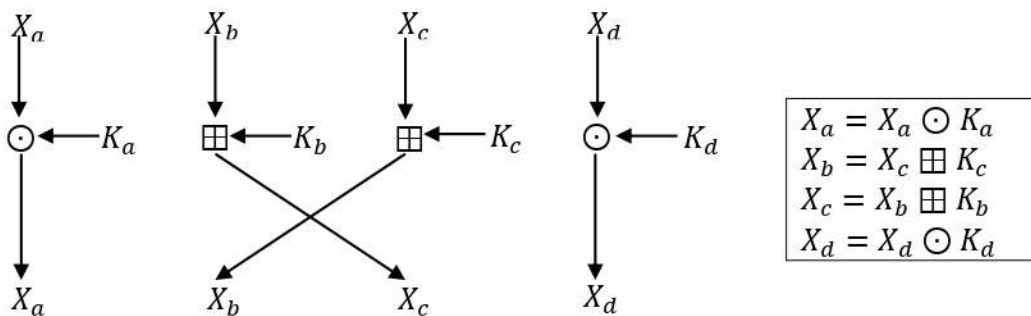


Fig: IDEA Odd Round

**Even Round**

- There are four input 16-bit sub-blocks ( $X_a, X_b, X_c, X_d$ ) from the previous round.
- Uses 2 16-bit keys ( $K_e$  &  $K_f$ ).

It works as follows:

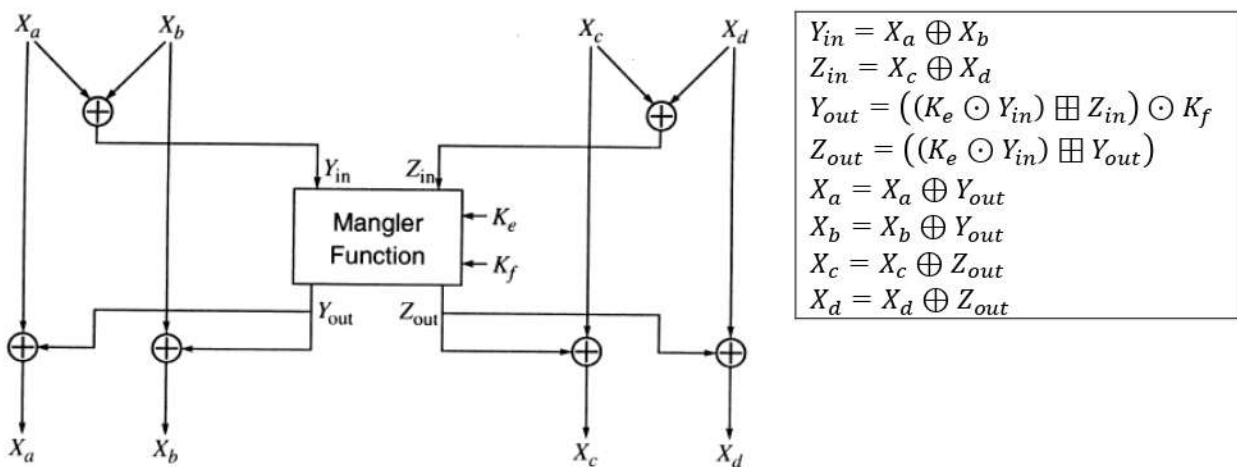


Fig: IDEA Even Round

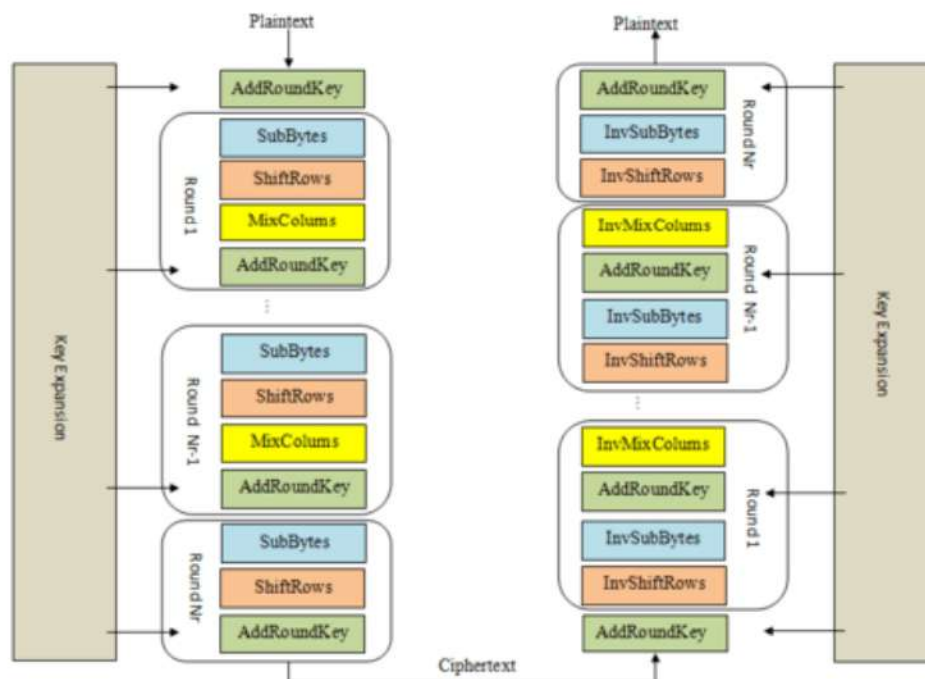


## Advance Encryption Standard (AES)

- AES is a block cipher which operates on block size of 128 bits for both encrypting as well as decrypting.
- Three key lengths are available: 128, 192, or 256 bits (16, 24, or 32 bytes)
- The number of rounds performed by the algorithm strictly depends on the size of key.

| Key Size (in bits) | Rounds |
|--------------------|--------|
| 128                | 10     |
| 192                | 12     |
| 256                | 14     |

- Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.
- Each round consists of four functions:
  1. Sub Bytes
  2. Shift Rows
  3. Mix Columns, not applied in last round.
  4. Add Round Key



**Fig: Encryption/Decryption Rounds in AES**

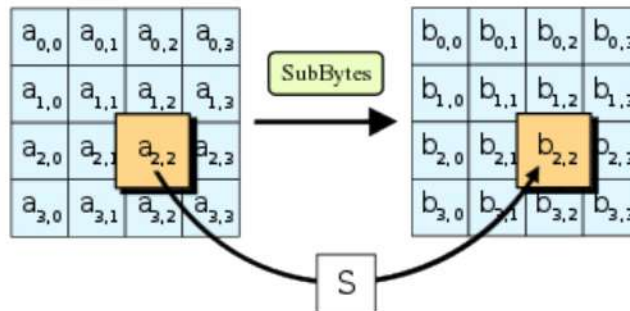
AES treats the 128-bits block (16 bytes) as a  $4 \times 4$  byte array, called state matrix.

|          |          |          |          |
|----------|----------|----------|----------|
| $a_{00}$ | $a_{01}$ | $a_{02}$ | $a_{03}$ |
| $a_{10}$ | $a_{11}$ | $a_{12}$ | $a_{13}$ |
| $a_{20}$ | $a_{21}$ | $a_{22}$ | $a_{23}$ |
| $a_{30}$ | $a_{31}$ | $a_{32}$ | $a_{33}$ |

All the four AES operations are applied on the matrices further. These operations can be described as:

**1. Sub Bytes**

In the SubBytes step, each byte in the state is replaced with the corresponding S-box value,  $S$ ;  $b_{ij} = S(a_{ij})$ .



The S-box is a special lookup table which is constructed by Galois fields. The generation algorithm used in this algorithm is  $GF(2^8)$  i.e 256 values are possible. The elements of S-box are written in hexadecimal system.

AES S-box lookup table appears as shown below:

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F  |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

For E.g.  $S(3C) = EB$  since  $EB$  appears in row 3 and column  $C$  of above given table.

For decryption, the inverse of SubBytes (InvSubBytes) is the same operation using the inversed S-box, which is also pre-calculated.

## 2. Shift Rows

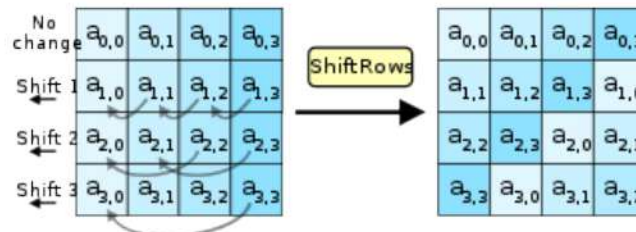
In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

1<sup>st</sup> row: no shift.

2<sup>nd</sup> row: a 1-byte circular left shift.

3<sup>rd</sup> row: a 2-byte circular left shift.

4<sup>th</sup> row: a 3-byte circular left shift.

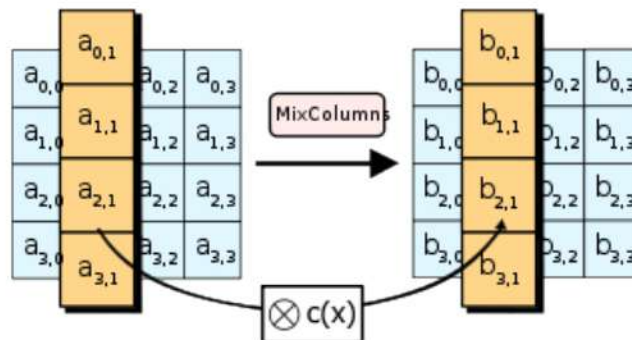


The inverse of Shift Row is the same cyclically shift but to the right.

## 3. Mix Columns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Each column of the state is multiplied with a fixed matrix. The multiplication is field multiplication in Galois field.



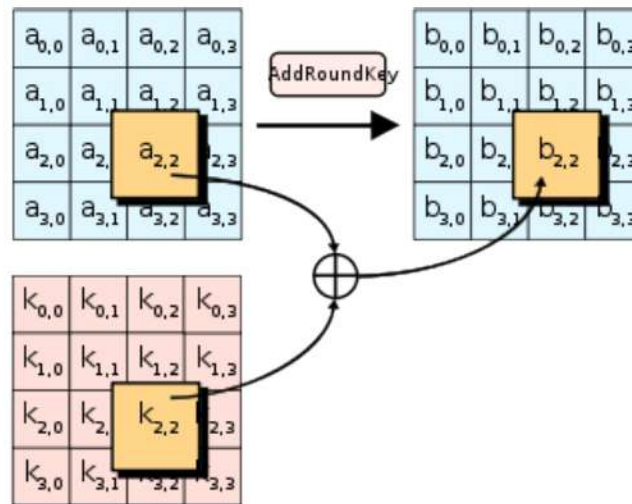
MixColumn matrix used for 128-bit key:

$$\begin{bmatrix} 2 & 31 & 1 \\ 1 & 23 & 1 \\ 1 & 12 & 3 \\ 3 & 11 & 2 \end{bmatrix}$$

## 4. Add Round Key

In the AddRoundKey step, each byte of the state is combined with a corresponding byte of the round subkey using the XOR operation ( $\oplus$ ).

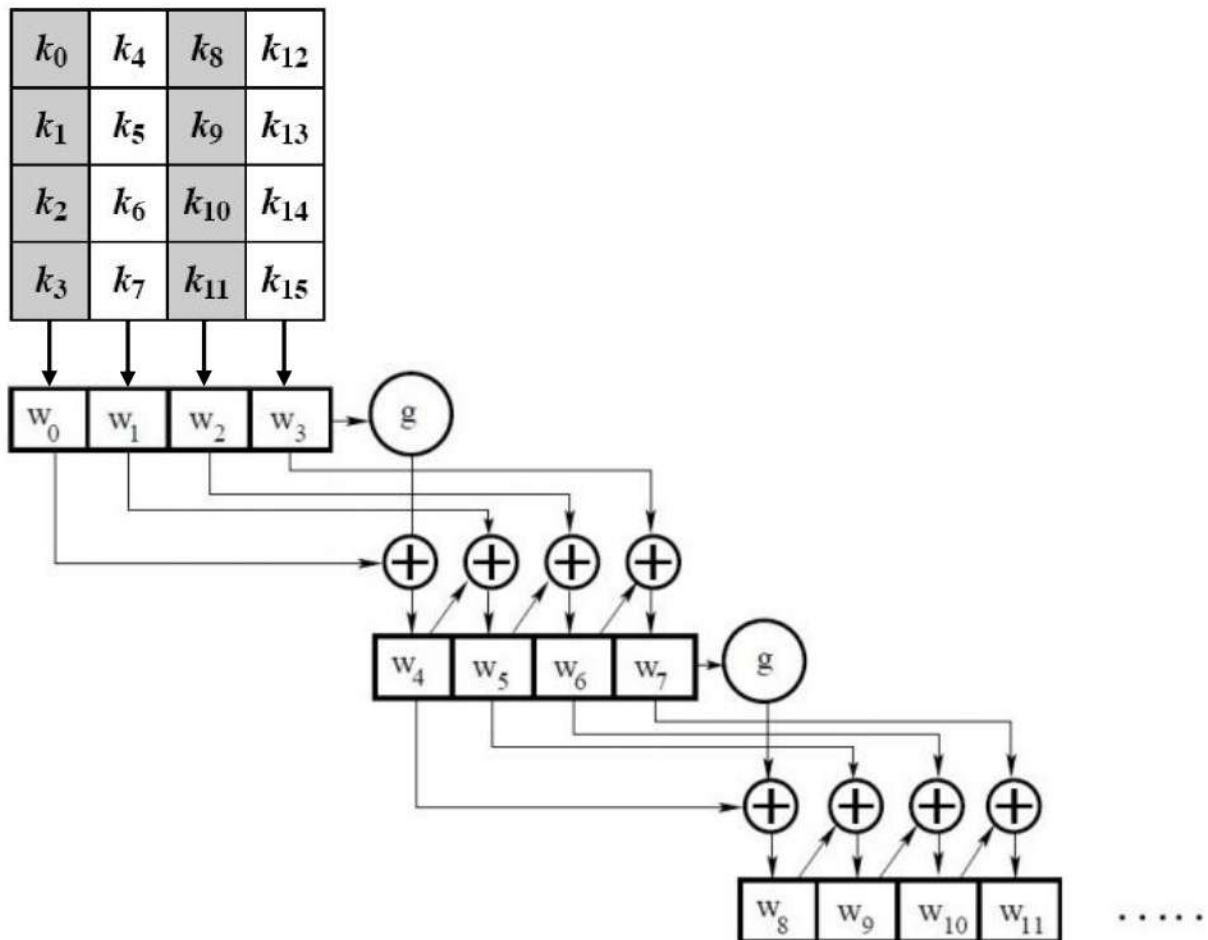




**AES Key Expansion**

To create round keys for each round, AES uses a key expansion process. If the number of rounds is  $N_r$ , the key expansion routines creates  $N_r + 1$  128-bit round keys for one single 128-bit cipher key.

It takes 128 bits (16-bytes) key and expands into array of 44 32-bit words. For the purpose of adding the key to state, each word is considered as column matrix.



**Fig: Key expansion in AES**



| Round     | Words      |              |              |              |
|-----------|------------|--------------|--------------|--------------|
| Pre-round | $w_0$      | $w_1$        | $w_2$        | $w_3$        |
| 1         | $w_4$      | $w_5$        | $w_6$        | $w_7$        |
| 2         | $w_8$      | $w_9$        | $w_{10}$     | $w_{11}$     |
| ...       | ...        |              |              |              |
| $N_r$     | $w_{4N_r}$ | $w_{4N_r+1}$ | $w_{4N_r+2}$ | $w_{4N_r+3}$ |

Calculation of  $g$  function includes following 3 processes:

1. **RotWord** performs a one byte circular left shift on a word. For e.g.  
 $RotWord[b_0, b_1, b_2, b_3] \rightarrow [b_1, b_2, b_3, b_0]$
2. **SubWord** performs a byte substitution on each byte of input word using S-box.
3. The result of step 1 and 2 is XORed with  $RCon[j]$ - the round constant.

### Round Constant (RCon):

- The round constant is a word in which the three rightmost bytes are always 0.
- It is different for each round and defined as:

$$RCon[j] = (RCon[j], 0, 0, 0)$$

Where,  $RCon[1] = 1$ ,  $RCon[j] = 2 * RCon[j - 1]$

| Round | Constant (RCon)         | Round | Constant (RCon)         |
|-------|-------------------------|-------|-------------------------|
| 1     | $(01\ 00\ 00\ 00)_{16}$ | 6     | $(20\ 00\ 00\ 00)_{16}$ |
| 2     | $(02\ 00\ 00\ 00)_{16}$ | 7     | $(40\ 00\ 00\ 00)_{16}$ |
| 3     | $(04\ 00\ 00\ 00)_{16}$ | 8     | $(80\ 00\ 00\ 00)_{16}$ |
| 4     | $(08\ 00\ 00\ 00)_{16}$ | 9     | $(1B\ 00\ 00\ 00)_{16}$ |
| 5     | $(10\ 00\ 00\ 00)_{16}$ | 10    | $(36\ 00\ 00\ 00)_{16}$ |

### Pseudocode for key expansion:

```

KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)    w[i] = (key[4*i], key[4*i+1],
                                   key[4*i+2],
                                   key[4*i+3]);

    for (i = 4; i < 44; i++)
    {
        temp = w[i - 1];
        if (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                               ⊕ Rcon[i/4];
        w[i] = w[i-4] ⊕ temp
    }
}

```

**Difference Between AES and DES**

| <b>AES</b>   | <b>DES</b>  |
|--|---|
| AES stands for Advanced Encryption Standard.   | DES stands for Data Encryption Standard.  |
| Block Size is 128 bits.  | Block size is 64 bits.  |
| Key length can be of 128-bits, 192-bits and 256-bits.  | Key length is 56 bits in DES.   |
| AES divides plaintext into 128-bit block each and treats each block as a 4 x 4 array. The block is then encrypted using one of the three different key lengths, 128, 192 & 256 bits. | DES divides plaintext message 64-bit block each and encrypts using 56-bit key.                |
| Number of rounds depends on key length : 10(128-bits), 12(192-bits) or 14(256-bits)  | DES involves 16 rounds of identical operations  |
| AES structure is based on substitution-permutation network.  | DES structure is based on feistel network.  |
| AES is faster.   | DES is comparatively slower.  |
| AES is more secure than DES.   | DES is less secure  |
| The rounds in AES are: Byte Substitution, Shift Row, Mix Column and Key Addition.  | The rounds in DES are: Expansion, XOR operation with round key, Substitution and Permutation. |
| AES was designed by Vincent Rijmen and Joan Daemen.  | DES was designed by IBM.  |

## Modes of Block Cipher Encryptions

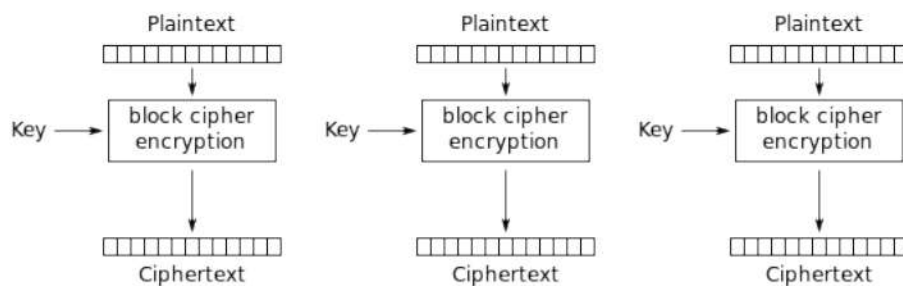
### 1. Electronic Code Book (ECB) Mode

In ECB mode, the message is divided into blocks of given size and each block of plaintext/ciphertext is encrypted/decrypted separately using the same key.

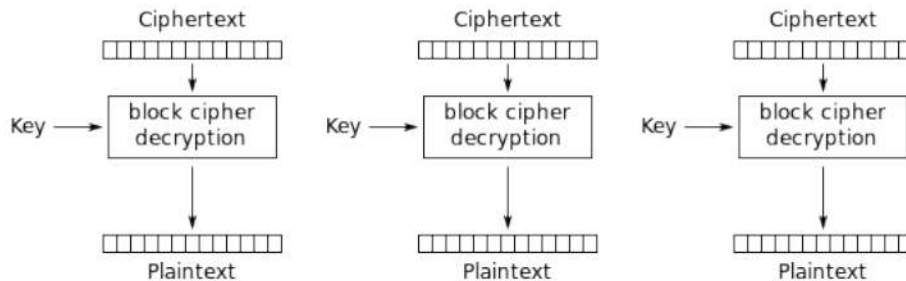
Encryption:  $C_i = E(P_i, key)$

Decryption:  $P_i = D(C_i, key)$

The same bit block of plaintext appears more than once in the message, it always produces the same ciphertext.



Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

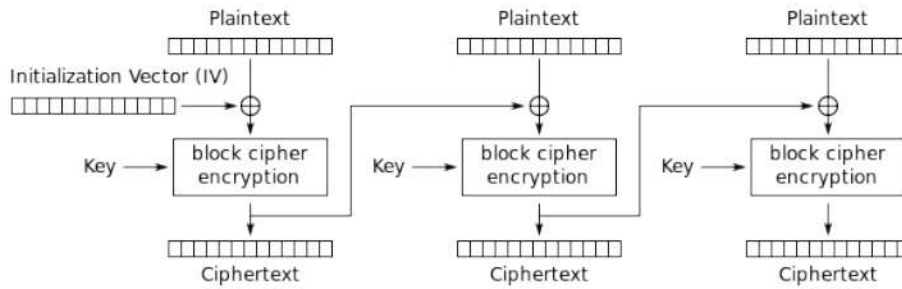
### 2. Cipher Block Chaining (CBC) mode

In CBC mode, each block of current plaintext is XORed with the previous ciphertext block before being encrypted with key. This way, each ciphertext block depends on all plaintext blocks processed up to that point. To make each message unique, an initialization vector (IV) must be used in first block.

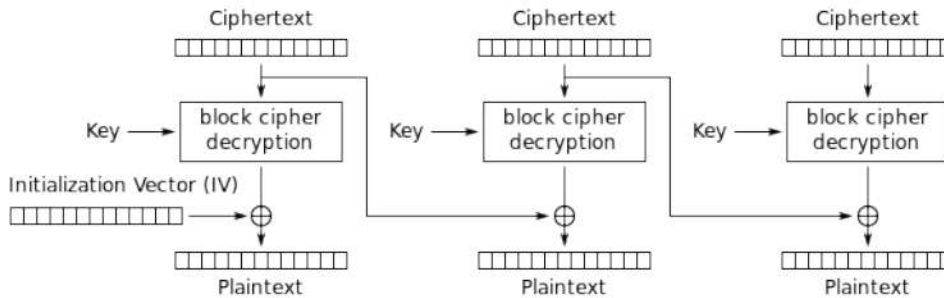
Decryption involves decrypting the current ciphertext block and then XORed with the previous ciphertext block.

Encryption:  $C_i = E(P_i \oplus C_{i-1}, key), C_0 = IV$

Decryption:  $P_i = D(C_i, key) \oplus C_{i-1}, C_0 = IV$



Cipher Block Chaining (CBC) mode encryption



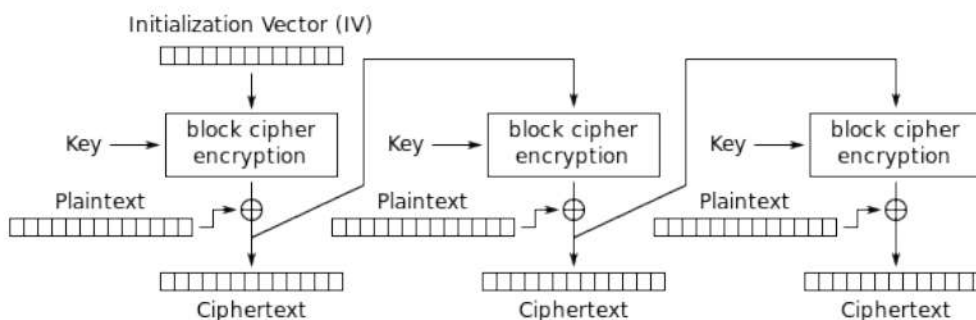
Cipher Block Chaining (CBC) mode decryption

### 3. Cipher Feedback (CFB) Mode

- A way of using a block cipher as a stream cipher.
- The shift register of block size maintains the current state of the cipher operation, initially set to some initialization vector (IV).
- The value of the shift register is encrypted using key K and the leftmost  $j$  bits of the output is XORed with  $j$ -bit plaintext  $P_i$  to produce  $j$ -bit ciphertext  $C_i$ .
- The value of the shift register is shifted left by  $j$  bits and the  $C_i$  is feedback to the rightmost  $j$ -bits of the shift register.
- Typically  $j = 8, 16, 32, \dots$
- 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.

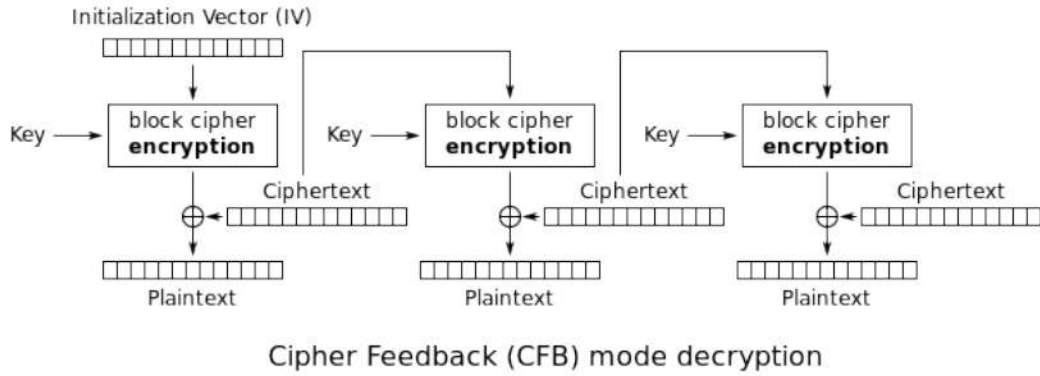
Encryption:  $C_i = E(C_{i-1}, key) \oplus P_i, C_0 = IV$

Decryption:  $P_i = E(C_{i-1}, key) \oplus C_i, C_0 = IV$



Cipher Feedback (CFB) mode encryption

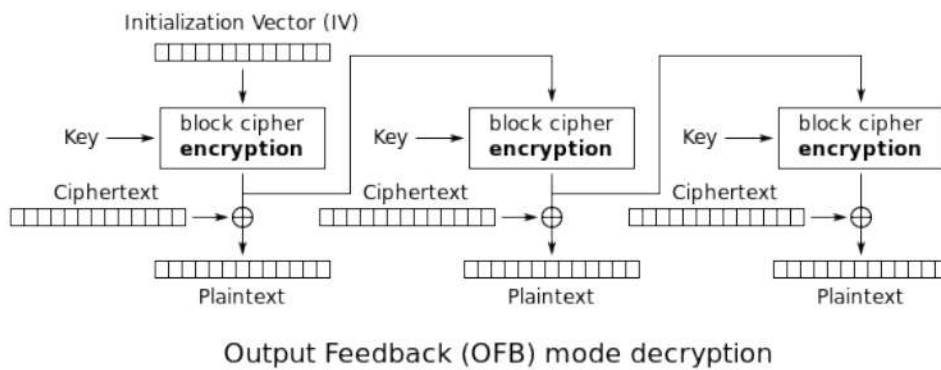
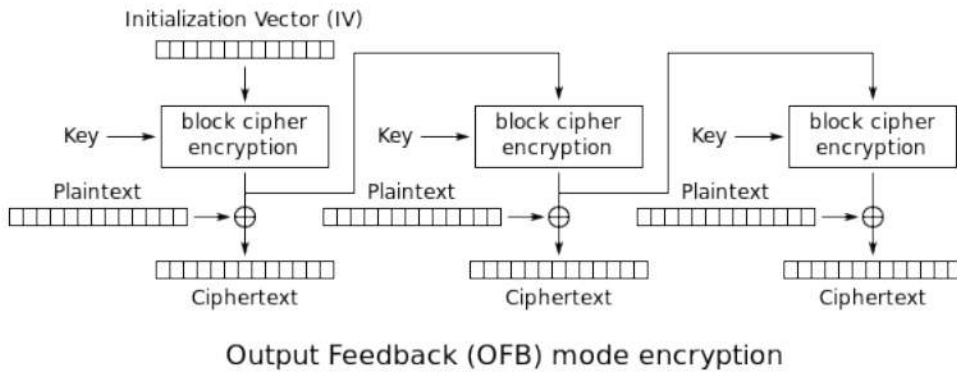




**4. Output Feedback (OFB) Mode**

It is similar to CFB mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output. In this output feedback mode, all bits of the block are send instead of sending selected  $j$  bits.

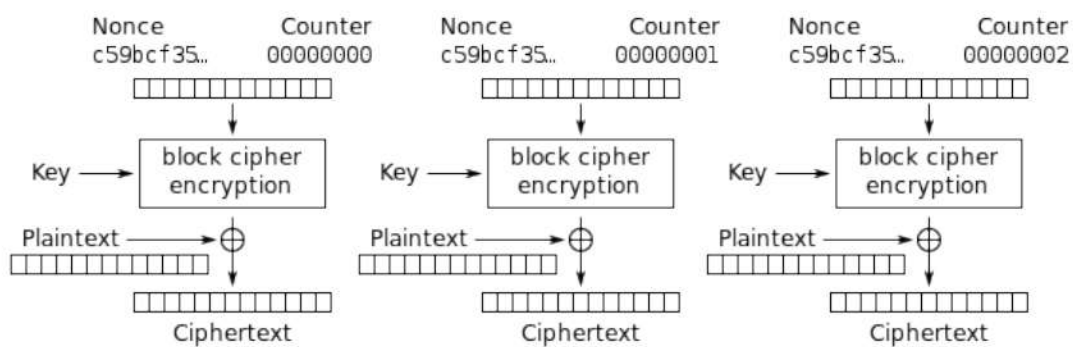
Encryption:  $C_j = P_j \oplus O_j$   
 Decryption:  $P_j = C_j \oplus O_j$   
 $O_j = E(I_j, key)$   
 $I_j = O_{j-1}$   
 $I_0 = IV$



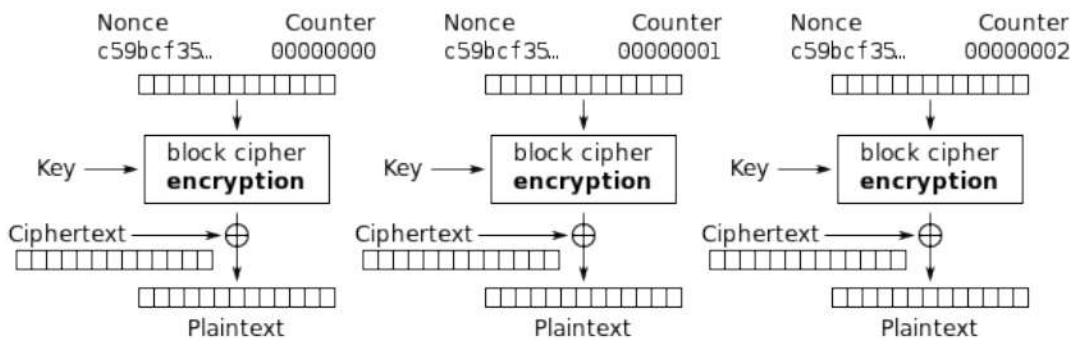
### 5. Counter (CTR) Mode

In this mode a counter, equal to the plaintext block size is used. The counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value (nonce + counter) and then incremented by 1 for each subsequent block. The nonce is a random number used for all blocks of an encryption operation and the counter is exactly what it sounds like: a value that starts at zero for block zero and increments to one for block one and so on.

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.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

**Q. Consider the message blocks  $m_1, m_2, m_3$ . If the cipher block chaining mode DES encryption can be expressed as  $C_i = DES(m_i \oplus m_{i-1} \oplus C_{i-1})$ ;  $m_0 \oplus C_2 = IV$ . Now, write the expression for the DES decryption to extract each of the message blocks  $m_1, m_2, m_3$ .**

**Sol<sup>n</sup>:**

### Encryption

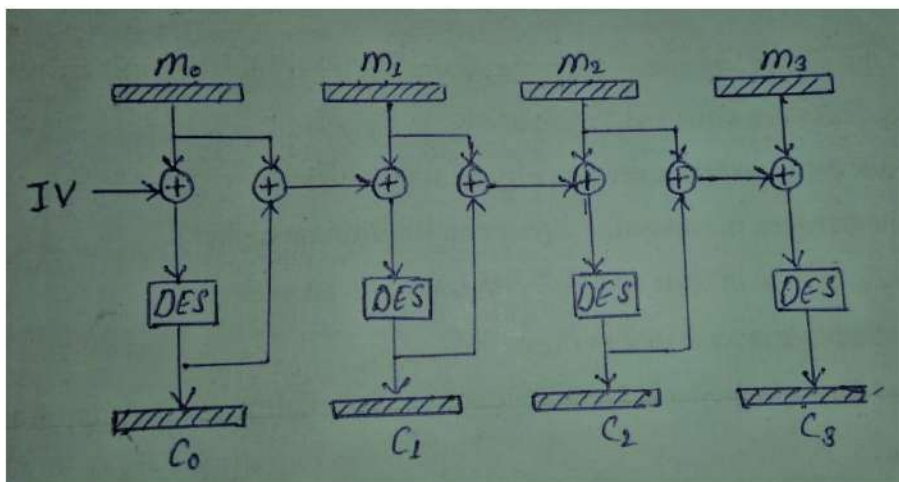
Given,

$$C_i = DES(m_i \oplus m_{i-1} \oplus C_{i-1})$$

$$C_1 = DES(m_1 \oplus m_0 \oplus C_0)$$

$$C_2 = DES(m_2 \oplus m_1 \oplus C_1)$$

$$C_3 = DES(m_3 \oplus m_2 \oplus C_2)$$



### Decryption

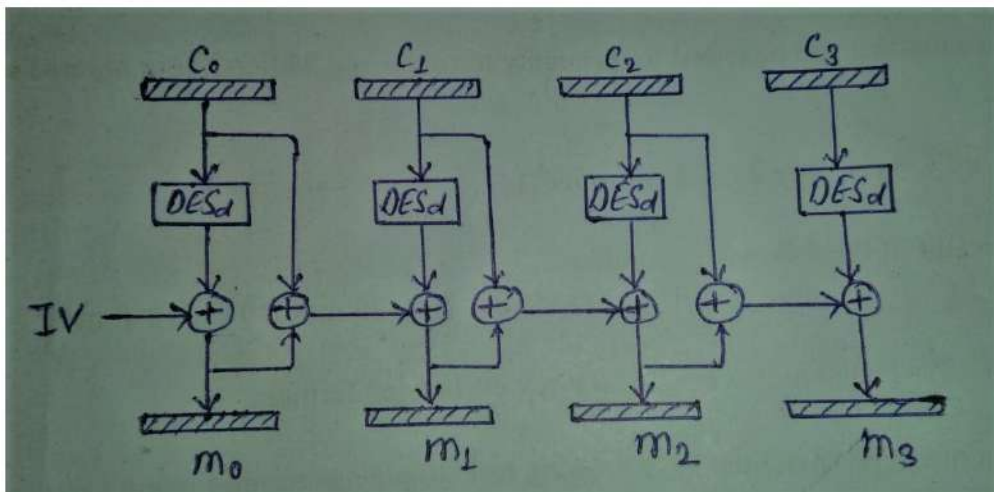
Expression for the decryption:

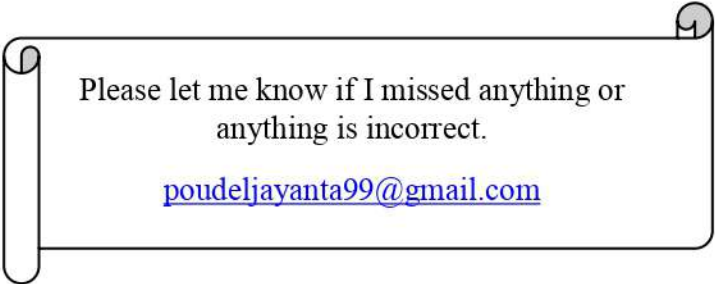
$$m_i = DES_d(C_i) \oplus m_{i-1} \oplus C_{i-1}$$

$$m_1 = DES_d(C_1) \oplus m_0 \oplus C_0$$

$$m_2 = DES_d(C_2) \oplus m_1 \oplus C_1$$

$$m_3 = DES_d(C_3) \oplus m_2 \oplus C_2$$





Please let me know if I missed anything or  
anything is incorrect.

[poudeljayanta99@gmail.com](mailto:poudeljayanta99@gmail.com)