UNIT- II:

Syllabus:

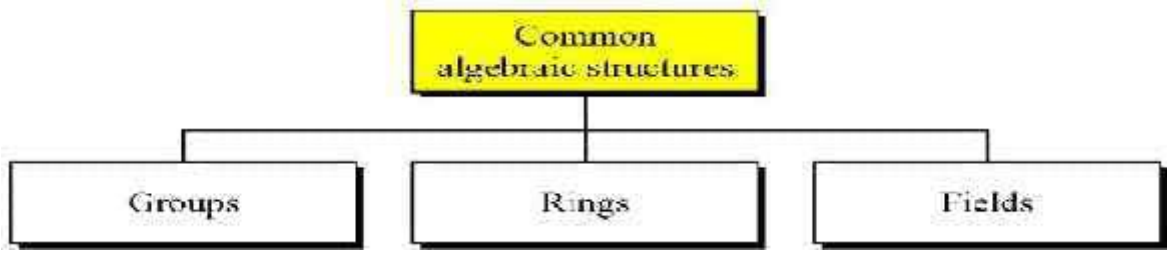
Symmetric Encryption: Mathematics of Symmetric Key Cryptography, Introduction to Modern Symmetric Key Ciphers, Data Encryption Standard, Advanced Encryption Standard.

TOPIC 01:

Algebraic structures:

In previous chapter we discuss some set of numbers,such as Z, Z_n, Z_n^*, Zp, Zp^*.

- Cryptography requires sets of integers and specific operations that are defined for those sets.
- The combination of the set and the operations that are applied to that elements of the set is called an algebraic structure.
- In this we will define three common algebraic structures: groups,rings, and fields.



- **GROUPS:**

A Group (G) is a set of elements with a binary operation "•" that satisfies four properties(or axioms). A commutative group , also called an abelian group,is a group in which the operator satisfies the four properties for group plus an extra property, commutativity. The four properties for groups plus commutativity are defined as follows:

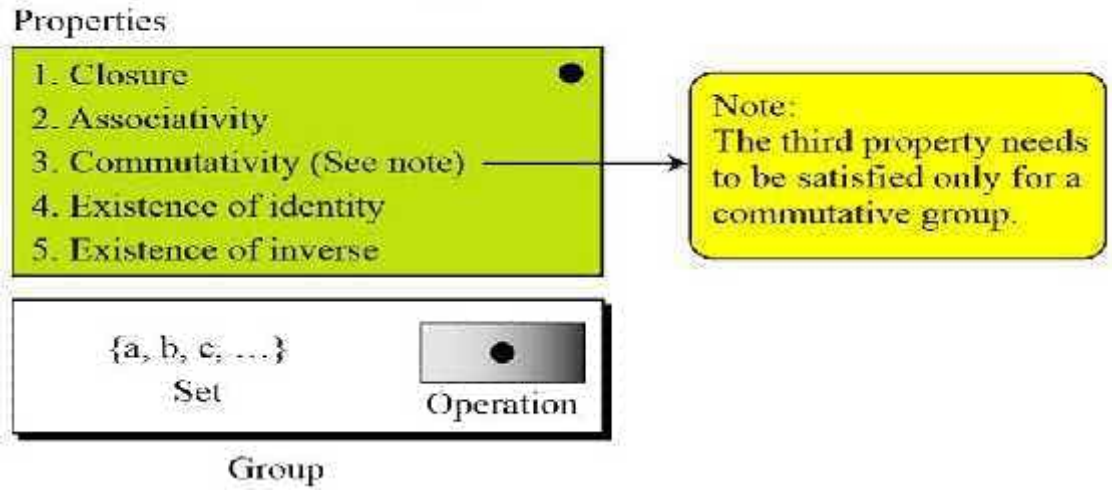
- **Closure:** if a and b are elements of G, then c= a •b is also an element of G. This means that the result of applying the operation on any two elements in the set is another element in the set.

- **Associativity:** If a,b,c are elements of G, then (a • b) •c = a • (b •c).

- **Commutativity:** For all a and b in G, we have (a • b)=(b.a).
- **Existence of Identity:** For all a in G, there exists an element e, called the identity element, such that e.a = a.e = a.

- **Existence of Inverse:** For each a in G,there exists an element a', called the inverse of a, such that a.a'=a'.a=e.

Figure 4.2 *Group*



Application:

- Although a group involves a single operation, the properties imposed on the operation allow the use of a pair of operaions as long as they are inverses of each other.
- For example, if the defined operation is addition, the group supports both addition and subtraction , because subtraction is addition using the additive inverse. This is also true for multiplication and division. However, a group can support only addition/subtraction or multiplication/division operations, but not the both at the same time.

Example 4.1 The set of residue integers with the addition operator, $G = \langle Z_n, + \rangle$, is a commutative group. We can perform addition and subtraction on the elements of this set without moving out of the set. Let us check the properties.

1. Closure is satisfied. The result of adding two integers in Z_n is another integer in Z_n.
2. Associativity is satisfied. The result of $4 + (3 + 2)$ is the same as $(4 + 3) + 2$.
3. Commutativity is satisfied. We have $3 + 5 = 5 + 3$.
4. The identify element is 0. We have $3 + 0 = 0 + 3 = 3$.
5. Every element has an additive inverse. The inverse of an element is its complement. For example, the inverse of 3 is -3 (n - 3 in Z_n) and the inverse of -3 is 3. The inverse allows us to perform subtraction on the set.

Finite Group:

A group is called a finite group if the set has a finite number of elements; otherwise, it is an infinite group.

Order of a group:

The order of a group , |G| , is the number of elements in the group. If the group is not finite, its order is infinite; if the group is finite, the order is finite.

Subgroups:

A subset H of a group G is a subgroup of G if H itself is a group with respect to the operation on

G. In other words , if G = < S, •> is a group , H = <T, •> is a group under the same operation, and T is a nonempty subset of S, then H is a subgroup of G. The above definition implies that:

1. if a and b are members of both groups, then c=a.b is also a member of both groups.
2. The group share the same identity element.
3. If a is a member of both groups, the inverse of a is also a member of both groups.
4. The group made of the identity element of G, H=< {e}, . >, is a subgroup of G.
5. Each group is a subgroup of itself.

Example 4.6 Is the group $H = \langle Z_{10}, + \rangle$ a subgroup of the group $G = \langle Z_{12}, + \rangle$?

Solution The answer is no. Although **H** is a subset of **G**, the operations defined for these two groups are different. The operation in **H** is addition modulo 10; the operation in **G** is addition modulo 12.

Cyclic Subgroups:

If a subgroup of a group can be generated using the power of an element, the subgroup is called the **cyclic subgroup.** The term *power* here means repeatedly applying the group operation to the element:

$a^n \rightarrow a \cdot a \ldots\ldots \cdot a$ (n times)

The set made from this process is referred to as <a>. Note that the duplicate elements must be discarded. Note also that a^0=e.

Example:

Four cyclic subgroups can be made from the group $G = \langle Z_6, + \rangle$. They are $H_1 = \langle \{0\}, + \rangle$, $H_2 = \langle \{0, 2, 4\}, + \rangle$, $H_3 = \langle \{0, 3\}, + \rangle$, and $H_4 = G$.

$0^0 \bmod 6 = 0$

$1^0 \bmod 6 = 0$
$1^1 \bmod 6 = 1$
$1^2 \bmod 6 = (1 + 1) \bmod 6 = 2$
$1^3 \bmod 6 = (1 + 1 + 1) \bmod 6 = 3$
$1^4 \bmod 6 = (1 + 1 + 1 + 1) \bmod 6 = 4$
$1^5 \bmod 6 = (1 + 1 + 1 + 1 + 1) \bmod 6 = 5$

$2^0 \bmod 6 = 0$
$2^1 \bmod 6 = 2$
$2^2 \bmod 6 = (2 + 2) \bmod 6 = 4$

$3^0 \bmod 6 = 0$
$3^1 \bmod 6 = 3$

$4^0 \bmod 6 = 0$
$4^1 \bmod 6 = 4$
$4^2 \bmod 6 = (4 + 4) \bmod 6 = 2$

$5^0 \bmod 6 = 0$
$5^1 \bmod 6 = 5$
$5^2 \bmod 6 = 4$
$5^3 \bmod 6 = 3$
$5^4 \bmod 6 = 2$
$5^5 \bmod 6 = 1$

Cyclic Groups:

- A Cyclic group is a group that is its own cyclic subgroup. The group **G** has a cyclic subgroup H_5 = G. This means that the group G is a cyclic group.

- In this case , the element that generates the cyclic subgroup can also generate the group itself.
- This element is referred to as a generator.
- If g is a generator, the element in a finite cyclic group can be written as, $\{e,g,g^2,g^3,\ldots\ldots,g^{n-1}\}$,where $g^n = e$

  Note that a cyclic group can have many generators.

Lagrange's Theorem:

- Lagrange's theorem relates the order of a group to the order of its subgroup.
- Assume that G is a group, and H is a subgroup of G. If the order of G and H are |G| and |H| ,respectively, then based on this theorem, |H| divides |G|.

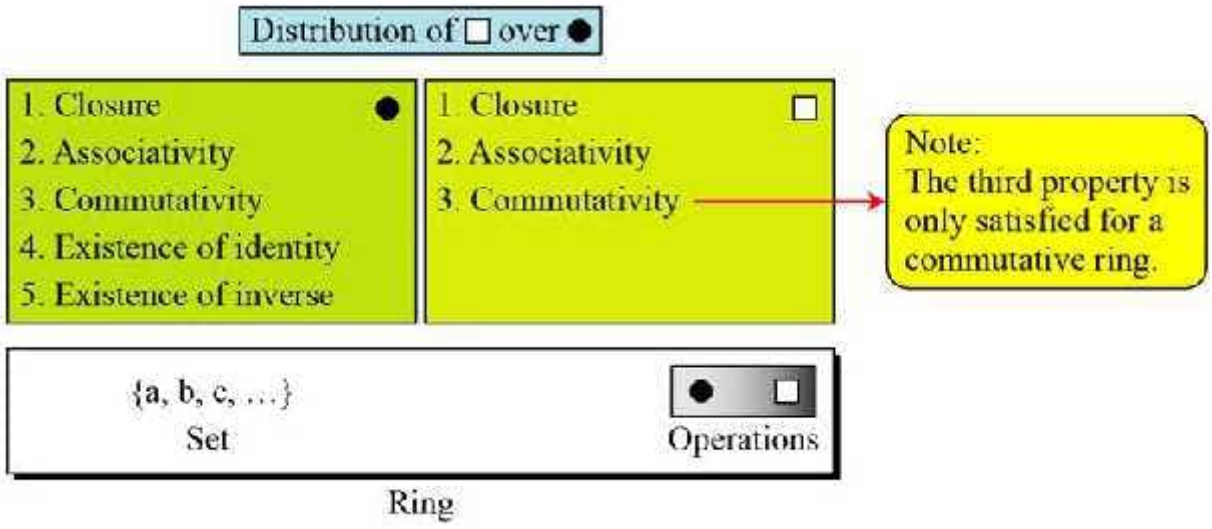
- In this example , H_1=<{0},+>,H_2 = <{0,2,4}, +>, H_3=<{0,3}, +>, then $|H_1|=1, |H_2|=3, |H_3|=2$.

Order of an element:

- The order of an element a in a group, ord(a),is the smallest integer n such that $a^n = e$.

➢ **Ring:**

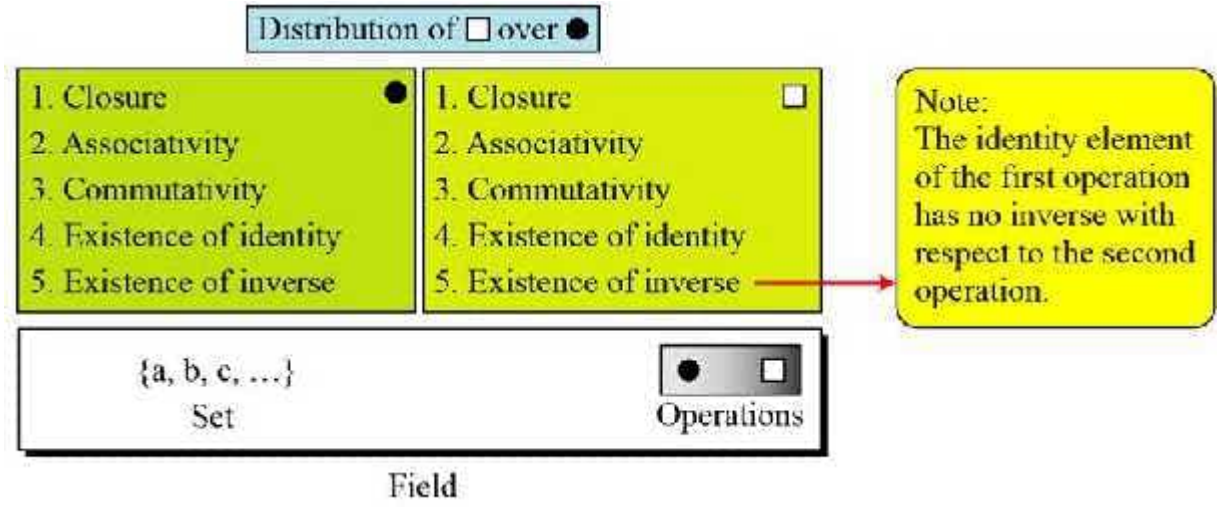
- A ring, denoted as R = <{ ……}, $\bullet$, □ **>** , is an algebraic structure with two operations.
- The first operation must satisfy all 5 properties required for an abelian group.
- The second operation must satisfy only the first two.
- In addition, the second operation must be distributed over the first.
- **Distributivity** means that for all a,b, and c elements of R, we have
  a □ (b ● c) = (a □ b) ●(a □c) and (a ● b) □ c =(a□ c) ● (b □ c).
- A **commutative ring** is a ring in which the commutative property is also satisfied for the second operation.



Application: A ring involves two operations. However the second operation can fail to satisfy the third and fourth properties.

- **Field:**

- A **field**, denoted by F= <{ ……}, ● , □ > is a commutative ring in which the second operation satisfies all five properties defined for the first operation except that the identity of the first operation has no inverse.



Application:

A field is a structure that supports two pairs of operations that we have used in mathematics: addition/subtraction and multiplication/division. There is an exception: division by zero is not allowed.

- **Finite fields:**

- Although we have fields of infinite order, only finite fields extensively used in cryptography. A finite field, a field with a finite number of elements,are very important structures in cryptography.

- Galios showed that for a field to be finiter,the number of elements should be p^n,where p is prime and n is a positive integer.
- The finite fields are usually called as Galios fields and denoted as **GF(p^n).**-

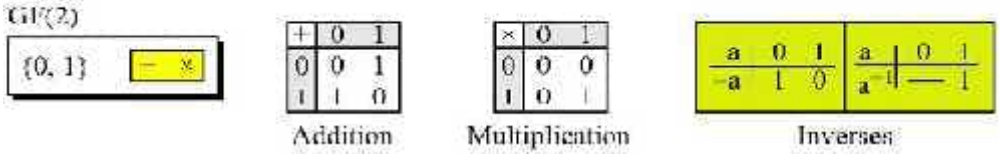
GF(P) Fields:

- When n=1, we have GF(p)field.The field can be the set Z_p,{0,1,2,…….p-1},with two arithmetic operations.

- Recall that each element has an additive inverse and that nonzero elements have a multiplicative inverse(no multiplicative inverse for 0)

A very common field in this category is GF(2) with the set {0, 1} and two operations, addition and multiplication, as shown in Figure 4.6.

Figure 4.6 *GF(2) field*

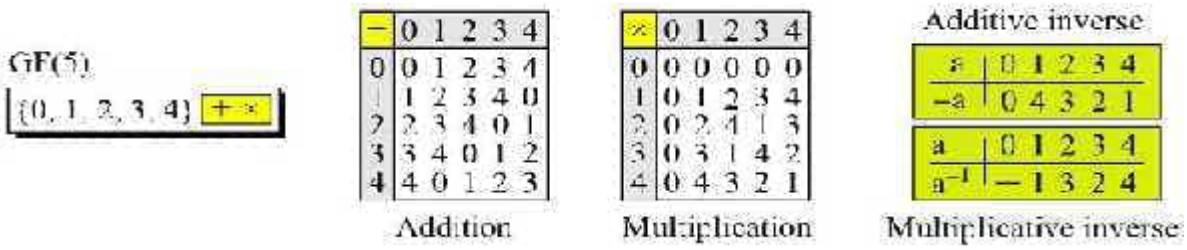


Example:1

There are several things to notice about this field.First, the set has only two elements, which are binary digits or bits(0 and 1) . second, the addition operation is actually the exclusive-or(XOR) operation we use on two binary digits.Third , the multiplication operation is the AND operation we use on two binary digits. Fourth, addition and subtraction are the same(XOR). Fifth, multiplication and division are the same(AND operation).

Example :2

- We can define GF(5) on the set Z_5 (5 is a prime) with addition and multiplication operators



Although we can use the extended Euclidean algorithm to find the multiplicative inverses of elements in GF(5), it is simpler to find each pair with the product equal to 1.They are (1,1),(2,3),(3,2),(4,4).

GF(p^n) Fields:

In addition to GF(p) fields,we have to know GF(p^n) fields in cryptography.

Algebraic Structure	*Supported Typical Operations*	*Supported Typical Sets of Integers*
Group	$(+ \ -)$ or $(\times \ \div)$	$\mathbf{Z}_n$ or $\mathbf{Z}_n^*$
Ring	$(+ \ -)$ and $(\times)$	$\mathbf{Z}$
Field	$(+ \ -)$ and $(\times \ \div)$	$\mathbf{Z}_p$

Summary:

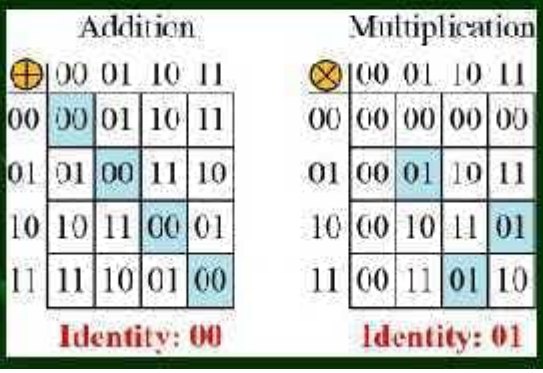
- **GF(2^n) FIELDS:**

In cryptography, we often need to use four operations (addition, subtraction, multiplication, and division). In other words, we need to use fields. We can work in GF(2^n) and uses a set of 2^n elements. The elements in this set are n-bit words.

1. We can use GF(p) with the set Zp, where p is the largest prime number less than 2^n. For example, if n=4, the largest prime less than 2^4 is 13.This means that we cannot use integers 13,14,15.
2. We can work in GF(2^n) and uses a set of 2^n elements. The elements in this setare n-bit words, for example, if n=3, the set is
   {000,001,010,011,100,101,110,111}

Let us define a GF(2^2) field in which the set has four 2-bit words: {00, 01, 10, 11}. We can redefine addition and multiplication for this field in such a way that all properties of these operations are satisfied, as shown in Figure 4.8.

Figure 4.8 *An example of GF(2^2) field*



Polynomials:

Although we can directly define the rules for addition and multiplication operations on n-bit words that satisfy the properties in GF(2^n). It is easier to work with a representation of n-bit words, a polynomial of degree n-1, A polynomial of degree n-1 is an expression of the form

$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum a_i x^i$

where x^i is called the ith term and a_i is called coefficient of the ith term. Although we are familiar with polynomials in algebra, to represent an n-bit word by a polynomial we need to follow some rules:

a. The power of x defines the position of the bit in the n-bit word. This means the leftmost bit is at position zero (related to x^0); the rightmost bit is at position $n - 1$ (related to x^{n-1}).
b. The coefficients of the terms define the value of the bits. Because a bit can have only a value of 0 or 1, our polynomial coefficients can be either 0 or 1.

Example 4.15 Figure 4.9 shows how we can represent the 8-bit word (10011001) using a polynomials.

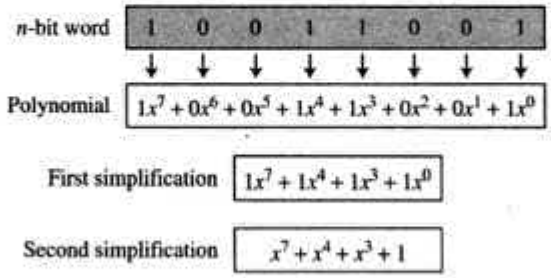


Fig. 4.9 *Representation of an 8-bit word by a polynomial*

Note that the term is totally omitted if the coefficient is 0, and the coefficient is omitted if it is 1. Also note that x^0 is 1.

Example 4.16 To find the 8-bit word related to the polynomial $x^5 + x^2 + x$, we first supply the omitted terms. Since $n = 8$, it means the polynomial is of degree 7. The expanded polynomial is

$$0x^7 + 0x^6 + 1x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 0x^0$$

This is related to the 8-bit word 00100110.

Operations Note that any operation on polynomials actually involves two operations: operations on coefficients and operations on two polynomials. In other words, we need to define two fields: one for the coefficients and one for the polynomials. Coefficients are made of 0 or 1; we can use the **GF**(2) field for this purpose. We discusses this field before (see Example 4.14). For the polynomials we need the field GF(2^n), which we will discuss shortly.

Polynomials representing n-bit words use two fields: GF(2) and GF(2^n).

Modulus Before defining the operations on polynomials, we need to talk about the modulus polynomials. Addition of two polynomials never creates a polynomial out of the set. However, multiplication of two polynomials may create a polynomial with degrees more than $n - 1$. This means

- We need to divide the result by a modulus and keep only the remainder, as in modular arithmetic.For the sets of polynomials in GF(2^n) , a group of polynomials of degree n is defined as the modulus.
- The modulus in this case acts as a prime polynomial, which means that no polynomials in the set can divide this polynomial.
- A prime polynomial cannot be factored into a polynomial with degree of less than n.such polynomials are referred to as irreducible polynomial.

Degree	*Irreducible Polynomials*
1	$(x + 1), (x)$
2	$(x^2 + x + 1)$
3	$(x^3 + x^2 + 1), (x^3 + x + 1)$
4	$(x^4 + x^3 + x^2 + x + 1), (x^4 + x^3 + 1), (x^4 + x + 1)$
5	$(x^5 + x^2 + 1), (x^5 + x^3 + x^2 + x + 1), (x^5 + x^4 + x^3 + x + 1),$ $(x^5 + x^4 + x^3 + x^2 + 1), (x^5 + x^4 + x^2 + x + 1)$

Addition: Now let us define the addition operation for polynomials with coeffieients in GF(2). Addition is very easy. We add the coefficients of the corresponding terms in GF(2).

Note that adding two polynomials of degree n-1 always create a polynomial with degree n-1,which means that we do not need to reduce the result using the modulus.

Example 4.17 Let us do $(x^5 + x^2 + x) \oplus (x^3 + x^2 + 1)$ in GF(2^8). We use the symbol $\oplus$ to show that we mean polynomial addition. The following shows the procedure:

$$\begin{array}{ll} 0x^7 + 0x^6 + 1x^5 + 0x^4 + 0x^3 + 1x^2 + 1x^1 + 0x^0 & \oplus \\ 0x^7 + 0x^6 + 0x^5 + 0x^4 + 1x^3 + 1x^2 + 0x^1 + 1x^0 & \\ \hline 0x^7 + 0x^6 + 1x^5 + 0x^4 + 1x^3 + 0x^2 + 1x^1 + 1x^0 & \rightarrow \quad x^5 + x^3 + x + 1 \end{array}$$

There is a short cut: keeps the uncommon terms and delete the common terms. In other words, x^5, x^3, x, and 1 are kept and x^2, which is common in the two polynomials, is deleted.

Additive Identity:

The additive identity in a polynomial is a zero polynomial because adding the polynomial with itself results in a zero polynomial.

Additive Inverse:

The additive inverse of a polynomial with coefficients in GF(2) is the polynomial itself. This means that the subtraction operation is the same as the addition operation.

Multiplication:

Multiplication in polynomial is the sum of the multiplication of each term of the first polynomial with each term of the second polynomial.we have to consider three points.

Example 4.19 Find the result of $(x^5 + x^2 + x) \otimes (x^7 + x^4 + x^3 + x^2 + x)$ in GF(2^8) with irreducible polynomial $(x^8 + x^4 + x^3 + x + 1)$. Note that we use the symbol $\otimes$ to show the multiplication of two polynomials.

Solution We first multiply the two polynomials as we have learned in algebra. Note that in this process, a pair of terms with equal power of x are deleted. For example, $x^9 + x^9$ is totally deleted because the result is a zero polynomial, as we discussed above.

$$P_1 \otimes P_2 = x^5(x^7 + x^4 + x^3 + x^2 + x) + x^2(x^7 + x^4 + x^3 + x^2 + x) + x(x^7 + x^4 + x^3 + x^2 + x)$$
$$P_1 \otimes P_2 = x^{12} + x^9 + x^8 + x^7 + x^6 + x^9 + x^6 + x^5 + x^4 + x^3 + x^8 + x^5 + x^4 + x^3 + x^2$$
$$P_1 \otimes P_2 = (x^{12} + x^7 + x^2) \bmod (x^8 + x^4 + x^3 + x + 1) = x^5 + x^3 + x^2 + x + 1$$

To find the final result, divide the polynomial of degree 12 by the polynomial of degree 8 (the modulus) and keep only the remainder. The process is the same as we have learned in algebra, but we need to remember that subtraction is the same as addition here. Figure 4.10 shows the process of division.

$$\begin{array}{r|l} & x^4 + 1 \\ \hline x^8 + x^4 + x^3 + x + 1 & x^{12} + x^7 + x^2 \\ & x^{12} + x^8 + x^7 + x^5 + x^4 \\ \hline & x^8 + x^5 + x^4 + x^2 \\ & x^8 + x^4 + x^3 + x + 1 \\ \hline \text{Remainder} & \boxed{x^5 + x^3 + x^2 + x + 1} \end{array}$$

Multiplicative Identity The multiplicative identity is always 1. For example, in GF(2^8), the multiplicative inverse is the bit pattern 00000001.

Multiplicative Inverse Finding the multiplicative inverse is a little more involved. The extended Euclidean algorithm must be applied to the modulus and the polynomial. The process is exactly the same as for integers.

Example 4.20 In GF(2^4), find the inverse of $(x^2 + 1)$ modulo $(x^4 + x + 1)$.

Solution We use the extended Euclidean algorithm as in Table 4.5:

Table 4.5 *Euclidean algorithm for Exercise 4.20*

q	r_1	r_2	r	t_1	t_2	t
$(x^2 + 1)$	$(x^4 + x + 1)$	$(x^2 + 1)$	(x)	(0)	(1)	$(x^2 + 1)$
(x)	$(x^2 + 1)$	(x)	(1)	(1)	$(x^2 + 1)$	$(x^3 + x + 1)$
(x)	(x)	(1)	(0)	$(x^2 + 1)$	$(x^3 + x + 1)$	(0)
	(1)	(0)		$(x^3 + x + 1)$	(0)	

This means that $(x^2 + 1)^{-1}$ modulo $(x^4 + x + 1)$ is $(x^3 + x + 1)$. The answer can be easily proved by multiplying the two polynomials and finding the remainder when the result is divided by the modulus.

$$[(x^2 + 1) \otimes (x^3 + x + 1)] \bmod (x^4 + x + 1) = 1$$

Example 4.21 In GF(2^8), find the inverse of (x^5) modulo ($x^8 + x^4 + x^3 + x + 1$).

Solution Use the Extended Euclidean algorithm as shown in Table 4.6:

Table 4.6 *Euclidean algorithm for Example 4.21*

q	r_1	r_2	r	t_1	t_2	t
(x^3)	$(x^8+x^4+x^3+x+1)$	(x^5)	(x^4+x^3+x+1)	(0)	(1)	(x^3)
$(x+1)$	(x^5)	(x^4+x^3+x+1)	(x^3+x^2+1)	(1)	(x^3)	(x^4+x^3+1)
(x)	(x^4+x^3+x+1)	(x^3+x^2+1)	(1)	(x^3)	(x^4+x^3+1)	$(x^5+x^4+x^3+x)$
(x^3+x^2+1)	(x^3+x^2+1)	(1)	(0)	(x^4+x^3+1)	$(x^5+x^4+x^3+x)$	(0)
	(1)	(0)		$(x^5+x^4+x^3+x)$	(0)	

This means that $(x^5)^{-1}$ modulo $(x^8 + x^4 + x^3 + x + 1)$ is $(x^5 + x^4 + x^3 + x)$. The answer can be easily proved by multiplying the two polynomials and finding the remainder when the result is divided by the modulus.

$$[(x^5) \otimes (x^5 + x^4 + x^3 + x)] \text{ modulo } (x^8 + x^4 + x^3 + x + 1) = 1$$

Multiplication Using a Computer Because of the division operation, there is an efficiency problem involved in writing a program to multiply two polynomials. The computer implementation uses a better algorithm, repeatedly multiplying a reduced polynomial by x. For example, instead of finding the result of $(x^2 \otimes P_2)$, the program finds the result of $(x \otimes (x \otimes P_2))$. The benefit of this strategy will be discussed shortly, but first let us use an example to show the process.

Example 4.22 Find the result of multiplying $P_1 = (x^5 + x^2 + x)$ by $P_2 = (x^7 + x^4 + x^3 + x^2 + x)$ in $GF(2^8)$ with irreducible polynomial $(x^8 + x^4 + x^3 + x + 1)$ using the algorithm described above.

Solution The process is shown in Table 4.7. We first find the partial result of multiplying x^0, x^1, x^2, x^3, x^4, and x^5 by P_2. Note that although only three terms are needed, the product of $x^m \otimes P_2$ for m from 0 to 5 because each calculation depends on the previous result.

Table 4.7 *An efficient algorithm for multiplication using polynomials (Example 4.22)*

Powers	*Operation*	*New Result*	*Reduction*
$x^0 \otimes P_2$		$x^7 + x^4 + x^3 + x^2 + x$	No
$x^1 \otimes P_2$	$x \otimes (x^7 + x^4 + x^3 + x^2 + x)$	$x^5 + x^2 + x + 1$	**Yes**
$x^2 \otimes P_2$	$x \otimes (x^5 + x^2 + x + 1)$	$x^6 + x^3 + x^2 + x$	No
$x^3 \otimes P_2$	$x \otimes (x^6 + x^3 + x^2 + x)$	$x^7 + x^4 + x^3 + x^2$	No
$x^4 \otimes P_2$	$x \otimes (x^7 + x^4 + x^3 + x^2)$	$x^5 + x + 1$	**Yes**
$x^5 \otimes P_2$	$x \otimes (x^5 + x + 1)$	$x^6 + x^2 + x$	No
$P_1 \times P_2 = (x^6 + x^2 + x) + (x^6 + x^3 + x^2 + x) + (x^5 + x^2 + x + 1) = x^5 + x^3 + x^2 + x + 1$			

The above algorithm has two benefits. First, multiplication of a polynomial by x can be easily achieved by one-bit shifting of the n-bit word; an operation provided by common programming languages. Second, the result needed to be reduced only if the polynomial maximum power is $n - 1$. In this case, reduction can be easily done by an XOR operation with the modulus because the highest power in the result is only 8. We can then design a simple algorithm to find each partial result:

1. If the most significant bit of the previous result is 0, just shift the previous result one bit to the left.
2. If the most significant bit of the previous result is 1,
   a. shift it one bit to the left, and
   b. exclusive-or it with the modulus without the most significant bit

Example 4.23 Repeat Example 4.22 using bit patterns of size 8.

Solution We have $P_1 = 000100110$, $P_2 = 10011110$, modulus = 100011010 (nine bits). We show the exclusive-or operation by $\oplus$. See Table 4.8.

Table 4.8 *An efficient algorithm for multiplication using n-bit words*

Powers	*Shift-Left Operation*	*Exclusive-Or*
$x^0 \otimes P_2$		10011110
$x^1 \otimes P_2$	00111100	(00111100) $\oplus$ (00011010) = **00100111**
$x^2 \otimes P_2$	01001110	**01001110**
$x^3 \otimes P_2$	10011100	10011100
$x^4 \otimes P_2$	00111000	(00111000) $\oplus$ (00011010) = 00100011
$x^5 \otimes P_2$	01000110	**01000110**
$P_1 \otimes P_2 = (00100111) \oplus (01001110) \oplus (01000110) = 00101111$		

In this case, we need only five shift-left operations and four exclusive-or operations to multiply the two polynomials. In general, a maximum of $n - 1$ shift-left operations and $2n$ exclusive-or operations are needed to multiply two polynomial of degree $n - 1$.

Multiplication of polynomials in GF(2^n) can be achieved using shift-left and exclusive-or operations.

Example 4.24 The GF(2^3) field has 8 elements. We use the irreducible polynomial ($x^3 + x^2 + 1$) and show the addition and multiplication tables for this field. We show both 3-bit words and the polynomials. Note that there are two irreducible polynomials for degree 3. The other one, ($x^3 + x + 1$), yields a totally different table for multiplication. Table 4.9 shows addition. The shaded boxes easily give us the additive inverses pairs.

Table 4.9 *Addition table for GF(2^3)*

⊕	000 (0)	001 (1)	010 (x)	011 ($x+1$)	100 (x^2)	101 x^2+1	110 (x^2+x)	111 (x^2+x+1)
000 (0)	000 (0)	001 (1)	010 (x)	011 ($x+1$)	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)
001 (1)	001 (1)	000 (0)	011 ($x+1$)	010 (x^2)	101 (x^2+1)	100 (x^2+x)	111 (x^2+x+1)	110 (x^2+x)
010 (x)	010 (x)	011 ($x+1$)	000 (0)	001 (1)	110 (x^2+x)	111 (x^2+x+1)	100 (x^2+x)	101 (x^2+1)
011 ($x+1$)	011 ($x+1$)	010 (x)	001 (1)	000 (0)	111 (x^2+x+1)	110 (x^2+x)	101 (x^2+1)	100 (x^2)
100 (x^2)	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)	000 (0)	001 (1)	010 (x)	011 ($x+1$)
101 (x^2+1)	101 (x^2+1)	100 (x^2)	111 (x^2+x+1)	110 (x^2+x)	001 (1)	000 (0)	011 ($x+1$)	010 (x)
110 (x^2+x)	110 (x^2+x)	111 (x^2+x+1)	100 (x^2)	101 (x^2+1)	010 (x)	011 ($x+1$)	000 (0)	001 (1)
111 (x^2+x+1)	111 (x^2+x+1)	110 (x^2+x)	101 (x^2+1)	100 (x^2)	011 ($x+1$)	010 (x)	001 (1)	000 (0)

Multiplication using a generator:

- Sometimes it is easier to define the elements of GF(2^n) using a generator. In this field with the irreducible polynomial f(x), an element in the field, a, must satisfy the relation f(a)=0.

Inverses

Finding inverses using the above representation is simple.

Additive Inverses The additive inverse of each element is the element itself because addition and subtraction in this field are the same: $-g^3 = g^3$

Multiplicative Inverses Finding the multiplicative inverse of each element is also very simple. For example, we can find the multiplicative inverse of g^3 as shown below:

$$(g^3)^{-1} = g^{-3} = g^{12} = g^3 + g^2 + g + 1 \rightarrow (1111)$$

Note that the exponents are calculated modulo $2^n - 1$, 15 in this case. Therefore, the exponent –3 mod 15 = 12 mod 15. It can be easily proved that g^3 and g^{12} are inverses of each other because $g^3 \times g^{12} = g^{15} = g^0 = 1$.

Operations The four operations defined for the field can also be performed using this representation.

Addition and Subtraction Addition and subtraction are the same operation. The intermediate results can be simplified as shown in the following example.

Example 4.26 The following show the results of addition and subtraction operations:

a. $g^3 + g^{12} + g^7 = g^3 + (g^3 + g^2 + g + 1) + (g^3 + g + 1) = g^3 + g^2 \rightarrow (1100)$
b. $g^3 - g^6 = g^3 + g^6 = g^3 + (g^3 + g^2) = g^2 \rightarrow (0100)$

$\{0, g, g, g^2, \ldots, g^N\}$, where $N = 2^n - 2$

⊗	000 (0)	001 (1)	010 (x)	011 ($x+1$)	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)
000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)
001 (1)	000 (0)	001 (1)	010 (x)	011 ($x+1$)	100 (x^2)	101 (x^2+1)	110 (x^2+x)	111 (x^2+x+1)
010 (x)	000 (0)	010 (x)	100 (x^2)	110 (x^2+x)	101 (x^2+1)	111 (x^2+x+1)	001 (1)	011 ($x+1$)

Multiplication and Division Multiplication is the addition of powers modulo $2^n - 1$. Division is multiplication using the multiplicative inverse.

Example 4.27 The following show the result of multiplication and division operations:

a. $g^9 \times g^{11} = g^{20} = g^{20 \bmod 15} = g^5 = g^2 + g \rightarrow (0110)$

b. $g^3 / g^8 = g^3 \times g^7 = g^{10} = g^2 + g + 1 \rightarrow (0111)$

Conclusion

The finite field GF(2^n) can be used to define four operations of addition, subtraction, multiplication and division over n-bit words. The only restriction is that division by zero is not defined. Each n-bit word can also be represented as a polynomial of degree $n - 1$ with coefficients in GF(2), which means that the operations on n-bit words are the same as the operations on this polynomial. To make it modular, we need to define an irreducible polynomial of degree n when we multiply two polynomials. The extended Euclidean algorithm can be applied to polynomials to find the multiplicative inverses.

TOPIC 02:

- **Introduction to modern block ciphers:**

A symmetric-key **modern block cipher** encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key. The decryption algorithm must be the inverse of the encryption algorithm, and both operations must use the same secret key so that Bob can retrieve the message sent by Alice. Figure 5.1 shows the general idea of encryption and decryption in a modern block cipher.

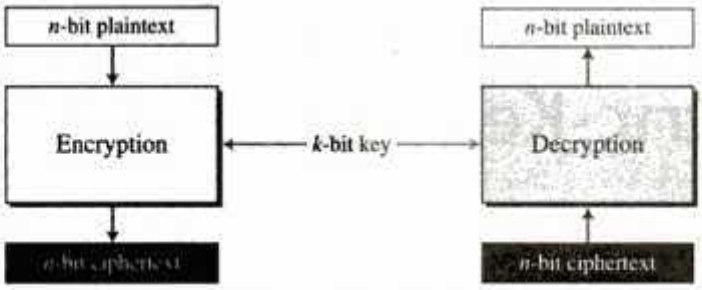


Fig. 5.1 *A modern block cipher*

If the message has fewer than n bits, padding must be added to make it an n-bit block; if the message has more than n bits, it should be divided into n-bit blocks and the appropriate padding must be added to the last block if necessary. The common values for n are 64, 128, 256, or 512 bits.

- **Substitution or Transposition:-**

A modern block cipher can be designed to act as a substitution cipher or a transposition cipher. This is the same idea as is used in traditional ciphers, except that the symbols to be substituted or transposed are bits instead of characters.

If the cipher is designed as a substitution cipher, a 1-bit or a 0-bit in the plaintext can be replaced by either a 0 or a 1. This means that the plaintext and the ciphertext can have a different number of 1's. A 64-bit plaintext block of 12 0's and 52 1's can be encrypted to a ciphertext of 34 0's and 30 1's. If the cipher is designed as a transposition cipher, the bits are only reordered (transposed); there is the same number of 1's in the plaintext and in the ciphertext. In either case, the number of n-bit possible plaintexts or ciphertexts is 2^n, because each of the n bits in the block can have one of the two values, 0 or 1.

- **Block ciphers as permutation groups:-**

As we will see in later chapters, we need to know whether a modern block cipher is a group (see Chapter 4). To answer this question, first assume that the key is long enough to choose every possible mapping from the input to the output. Call this a full-size key cipher. In practice, however, the key is smaller; only some mappings from the input to the output are possible. Although a block cipher needs to have a key that is a secret between the sender and the receiver, there are also keyless components that are used inside a cipher.

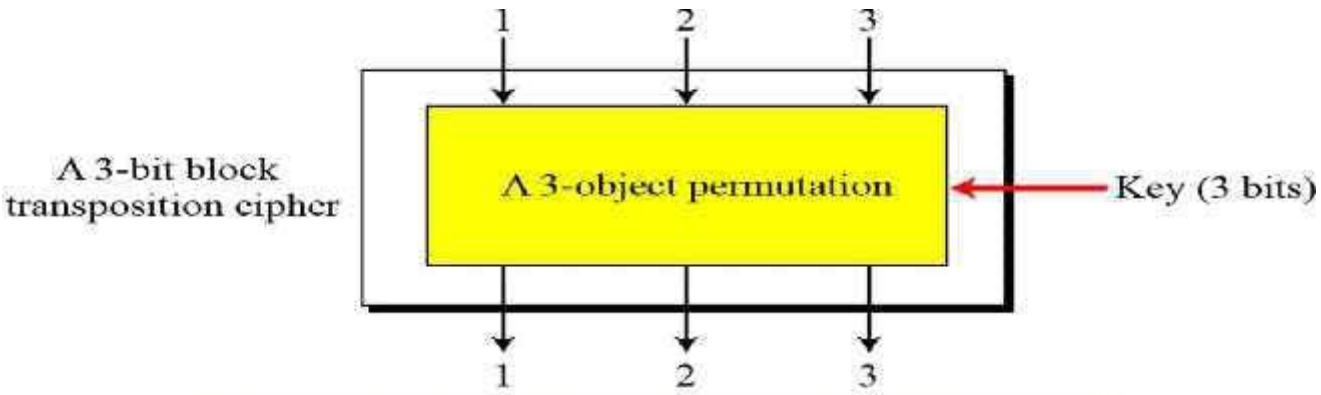
Full-Size Key Ciphers

Although full-size key ciphers are not used in practice, we first discuss this category to make the discussion of partial-size key ciphers understandable.

Full-Size Key Transposition Block Ciphers A full-size key transposition cipher only transposes bits without changing their values, so it can be modeled as an n-object permutation with a set of $n!$ permutation tables in which the key defines which table is used by Alice and Bob. We need to have $n!$ possible keys, so the key should have $\lceil \log_2 n! \rceil$ bits.

Example 5.3 Show the model and the set of permutation tables for a 3-bit block transposition cipher where the block size is 3 bits.

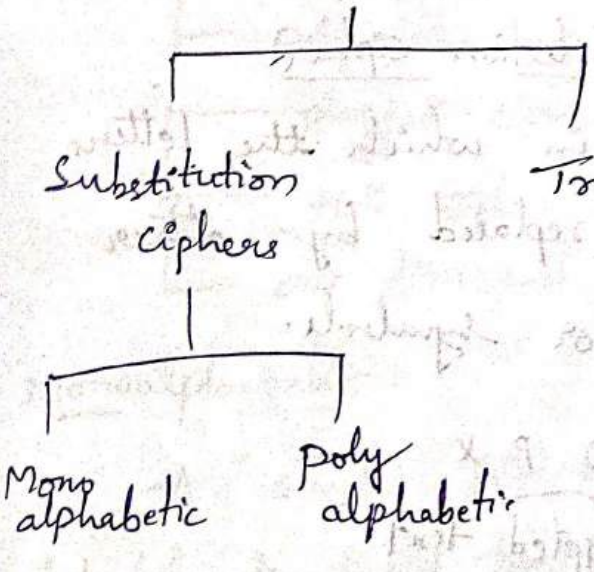
Solution The set of permutation tables has $3! = 6$ elements, as shown in Fig. 5.2. The key should be $\lceil \log_2 6 \rceil = 3$ bits long. Note that, although a 3-bit key can select $2^3 = 8$ different mappings, we use only 6 of them.



The set of permutation tables with $3! = 6$ elements

Unit II: Symmetric Encryption: Mathematics of Symmetric Key Cryptography, Introduction to Modern Symmetric key ciphers, Data Encryption Standard, Advanced Encryption standard

Traditional ciphers categorized broadly into two categories.



→ Sequence of characters will be changed

Subetitution: going to substitute some set characters by another set of characters

Mono alphabetic - one character at a time

Poly alphabetic - multiple characters at a time

Substitution: replaces one symbol with another

In mono alphabetic substitutions

plaintext: a b c d e f g h i j k l m n o p q r s t u v w x y z

Cipher text: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

Symmetric encryption also referred to as Conventional Encryption.

It is of 2 types 1) Substitution technique
2) Transposition technique

These two techniques are the basic building blocks of all the encryption techniques.

Substitution technique / substitution cipher

It is the one in which the letters of the plain text are replaced by other letters & by numbers or symbols.

Eg: Name → I W P X
plain text Encrypted text

Transposition techniques / Transposition ciphers

In this technique there is no replacement / substitution

In this technique some sort of permutations are performed on the plaintext letters i.e., it reorders the symbols

i.e., rearrangement of the letters of the plain text

Eg: NAME → EAMN or
AEMN or MNEA etc

Transposition ciphers

keyless keyed

Techniques
- Rail fence
- columnar transposition
- Double Transposition

Substitution Techniques
- Caesar (Mono)
- Monoalphabetic
- polyalphabetic
- playfair cipher
- Hill cipher
- one time pad.

Vigenere
Vernam ciphers

Monoalphabetic

A single cipher alphabet for each plain text alphabet is used throughout the process
i.e., fixed substitution.

if 'N' → I use 'x' then always y will use x only in place of N.

Eg MY NAME → NP OB N Z

In monoalphabetic cipher, relation between a character in the plaintext to a symbol in cipher text is always one to one.

Polyalphabetic

* There is no fixed substitutions
* Each occurence of character may have a different substitutions i.e., we can use more than 1 substitution for the same letter

Eg: MY NAME → NP OBXZ.

The relationship between a character in the plain text to a character in the ciphertext is one to many

Caesar cipher

It is also called shift cipher / additive cipher Each letter in the plaintext is replaced by a letter corresponding to a no. of shifts in the alphabet.

It is a Monoalphabetic caesar cipher and one of the simplest earliest method of encryption technique.

Note: Julius caesar used an additive cipher to communicate with his officers. For this reason additive ciphers are sometimes called caesar ciphers. He used a key of 3 for Communication

Eg: plain → Meet me zebra

PHHW PH CMEUD.

Ciphertext

$$C = E(K, P) = (P + K) \bmod 26 \quad // \text{ Encryption}$$

(K: Key, P: plain; P: message)

For decryption:

$$P = D(K, C) = (C - K) \bmod 26$$

(C: cipher)

Numerical value is assigned to each letter.

a	b	c	d	e	f	.....	x	y	z
0	1	2	3	4	5	.....	23	24	25

If the cryptanalist / attacker knows a ciphertext then he can apply brute force technique to find the plain text by using all the possible 25 keys.

Since it is a part of symmetric key encryption same key is used for encryption and decryption. $1 \le k \le 25$.

Eg: message → HELLO.

let Key = 4

$C(H) = (P + K) \bmod 26 = (7 + 4) \bmod 26 = 11 = L$

$C(E) = (P + K) \bmod 26 = (4 + 4) \bmod 26 = 8 = I$

$C(L) = (11 + 4) \bmod 26 = 15 = P$

$C(O) = (14 + 4) \bmod 26 = 18 = S$

Cipher → LIPPS.

Decryption: Cipher C = LIPPS

Key = 4

Now P = (C - K) mod 26)

P(L) = (11 - 4) mod 26 = 7 = H

P(I) = (8 - 4) mod 26 = 4 = E

P(P) = (15 - 4) mod 26 = 11 = O

P(S) = (18 - 4) mod 26 = 14 = O.

plain → HELLO.

Playfair cipher

It was invented in 1854 by charles wheatstone but was named after Lord playfair who promoted the use of cipher.

Cipher → Algorithm for encrypting and decrypting

Ciphertext → process which applies different types of algorithms to convert plaintext → coded text is called cipher text

Algorithm

1) create 5x5 matrix that is called grid of letters

2) The matrix is made by inserting the value of key and remaining alphabets into the matrix (row wise from left to right) where, letter I and J will be combined together.

3) Convert the text into pairs of alphabets

Eg: Heya → He ya.

(a) pair cannot be made with letters (same). Break the letters into single and add x to the previous letter

eg: Hello → He ll o
→ He lx lo

(b) If the letter is standing alone in the process of pairing then add 'z' with the letter

eg: Helloe → He lx lo e
He lx lo ez

Hexxoe → He xz xo ez

x was already there so we took z

4) Code will be formed using 3 rules

(i) If both the alphabets are in the same row, replace them with alphabets to their immediate right.

(ii) If both the alphabets are in the same column, replace them with alphabets immediately below them.

(iii) If not in same row / columns, replace them with alphabets in the same row respectively but at other pair of corners.

Example:

Key → Abhi

A	B	H	I/J	C
D	E	F	G	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

plain text { BM → ER
RW → BW

Same row { FG → GK.
VQ → QR.

Horizontal row. { QW → RV
KS → FU.

Vigenere ciphers. (Method 1)

Designed by Blaise de vigenere (16th century French mathematician).

It is a polyalphabetic substitution ciphers

The encryption is done using a (26 x 26) matrix or we a Table.

Method 1 → vigenere table → used to find cipher text

eg: plain text = Give money

key = Lock

Col + key

Sol

G	I	V	E	M	O	N	E	Y
L	O	C	K	L	O	C	K	L

Repeat the letters of the key so that the no of letters in P and K i.e, plaintext and key becomes equal

cipher → R W X O X C P O J.

Decryption

Row + key.

cipher → R W X O X C P O J

key → L O C K L O C K L

Rows in a table.

plain → G I V E M O N E Y.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R														
H	H	I	J	K	L	M	N	O	P																	
I	I	J	K	L	M	N	O	P																		
J	J	K	L	M	N	O	P	Q																		
K	K	L	M	N	O	P	Q	R																		
L	L	M	N	O	P	Q	R																			
M	M																									
N	N																									
O	O																									
P	P																									
Q	Q																									
R	R																									
S	S																									
T	T																									
U	U																									
V	V																									
W	W																									
X	X																									
Y	Y																									
Z	Z																									

Method 2

When the table is not given

1) Encryption

C_i (Cipher text) $= E_i = (P_i + K_i) \bmod 26$ // $E_i \rightarrow$ Encryption

2) Decryption.

$D_i = (E_i - K_i) \bmod 26$ // $D_i \rightarrow$ Decryption

Eg plain-text — she is listening.

Key — PASCAL

Key stream → (15, 0, 18, 2, 0, 11) the key stream is the repition of this initial key stream (as many times needed)

	18	7	4	8	18	11	8	18	19	4	13	8	13	6
plain	s	h	e	i	s	l	i	s	t	e	n	i	n	g
Key	15	0	18	2	0	11	15	0	18	2	0	11	13	6
e's value.	7	7	22	10	18	22	23	18	11	6	13	19	26	
Cipher text	H	H	W	K	S	W	X	S	L	G	N	T	C	G.
plain value	s	h	e	i	s	l	i	s	t	e	n	i	n	g.

- then (-8+26) mod 26

-4 + 26

22

Vernam cipher

→ used to encrypting alphabetic text.
→ simply a type of substitution cipher

In this we assign a number to each character of plain-text like (a=0, b=1, c=2, ... z=25)

Length of key used for encryption = length of plain text

eg: plain text → RAMSWAR,UPK
key → RANCHOBABA

Sol plain → 17 0 12 18 22 0 17 20 15 10
key → 17 0 13 2 7 14 1 0 1 0
(P+key) → 34 0 25 20 29 14 18 20 16 10
subst → 8 0 25 20 3 14 18 20 16 10.
cipher → I A Z U D O S U Q K.

Now for Decryption

cipher → 8 0 25 20 3 14 18 20 16 10
key 17 0 13 2 7 14 1 0 1 0
(-key) → -9 0 12 8 -4 0 17 20 15 10
plain 17 0 12 18 22 0 17 20 15 10

R A M S W A R U P K.

Hill Cipher

It is polyalphabetic cipher

→ Developed by Lester Hill in 1929

→ Encrypts a group of letters called polygraph like in playfair cipher, we saw it was encrypting a pair of letters which was called as a diagraph) so, here it can be a polygraph (diagraph, trigraph, etc).

This makes use of mathematics.

ABCDA → XMNPA.

To encrypt, $C = KP \bmod 26$.

K, P

Key plaintext.

Step 1: choose a key (key matrix must be a square matrix)

we can take any key

eg: $VIEW = \begin{bmatrix} V & I \\ E & W \end{bmatrix}_{2\times2} = \begin{bmatrix} 21 & 5 \\ 4 & 22 \end{bmatrix}_{2\times2}$

$Key = Quickness = \begin{bmatrix} Q & U & I \\ C & K & N \\ E & S & S \end{bmatrix}_{3\times3} = \begin{bmatrix} 16 & 20 & 8 \\ 2 & 10 & 13 \\ 4 & 18 & 18 \end{bmatrix}$

eg: plaintext = Attack

$Key = \begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix}$

nxn matrix
nx1 vector

Since the key is 2x2 matrix, plain text should be converted into vectors of length 2

So, $\begin{bmatrix} A \\ T \end{bmatrix}_{2\times1} \begin{bmatrix} T \\ A \end{bmatrix}_{2\times1} \begin{bmatrix} C \\ K \end{bmatrix}_{2\times1}$.

Now encryption begins

so, 1st vector $\rightarrow \begin{bmatrix} A \\ T \end{bmatrix} = \begin{bmatrix} 0 \\ 19 \end{bmatrix}$

key $= \begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix}$

$C = KP \bmod 26$

$C = \begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 0 \\ 19 \end{bmatrix} \bmod 26$

$= \begin{bmatrix} 2\times0 + 3(19) \\ 3\times0 + 6(19) \end{bmatrix} \bmod 26$

$= \begin{bmatrix} 57 \\ 114 \end{bmatrix} \bmod 26.$

corresponding alphabets $= \begin{bmatrix} F \\ K \end{bmatrix}$.

2nd vector $\begin{bmatrix} T \\ A \end{bmatrix} = \begin{bmatrix} 19 \\ 0 \end{bmatrix}$ key $\begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix}$.

$C = KP \bmod 26.$

$C = \begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 19 \\ 0 \end{bmatrix} = \begin{bmatrix} 2\times19 + 3\times0 \\ 3\times19 + 6\times0 \end{bmatrix} \bmod 26$

$C = \begin{bmatrix} 38 \\ 57 \end{bmatrix} \bmod 26 = \begin{bmatrix} 12 \\ 5 \end{bmatrix} = \begin{bmatrix} M \\ F \end{bmatrix}.$

3rd vector similarly for $\begin{bmatrix} C \\ K \end{bmatrix} = \begin{bmatrix} I \\ O \end{bmatrix}$.

plain text : AttAck

cipher text = FKMFIO.

Hill cipher - Decryption

To encrypt $C = KP \bmod 26$

To decrypt

Find inverse of key matrix K^{-1}

Eg plain → ATTACK, key $K = \begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix}$

$$K^{-1} = \frac{1}{|K|} adj(K).$$

$C = FK\ MF\ IO$

determinant of matrix $d = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$

$$d = |ad - bc|$$

eg $d = \begin{vmatrix} 2 & 3 \\ 3 & 6 \end{vmatrix} = |12 - 9| = 3.$

Now find multiplicative inverse of determinant

i.e., $dd^{-1} \equiv 1 \bmod 26$. (identity matrix)

so, $3 * d^{-1} \equiv 1 \bmod 26.$ $\quad 1 \bmod 26 = 1$

so $d^{-1} = 9.$ $\quad 3 * d^{-1} \bmod 26$

Now we find adjacent of the matrix

Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, then $adj(A) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

Here $K = \begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix}$, $adj(K) = \begin{bmatrix} 6 & -3 \\ -3 & 2 \end{bmatrix}$

Before decryption we have to remove -ve values (add 26 to -ve no).

$$adj(K) = \begin{bmatrix} 6 & -3+26 \\ -3+26 & 2 \end{bmatrix} = \begin{bmatrix} 6 & 23 \\ 23 & 2 \end{bmatrix}$$

Now $K^{-1} = \frac{1}{|K|} adj(K)$.

$$= 9\begin{bmatrix} 6 & 23 \\ 23 & 2 \end{bmatrix} = \begin{bmatrix} 54 & 207 \\ 207 & 18 \end{bmatrix}$$

$$K^{-1} = \begin{bmatrix} 54 & 207 \\ 207 & 18 \end{bmatrix} \bmod 26 = \begin{bmatrix} 2 & 25 \\ 25 & 18 \end{bmatrix}$$

$$K^{-1} = \begin{bmatrix} 2 & 25 \\ 25 & 18 \end{bmatrix}$$

Now, we will decrypt.

Cipher = FK MF IO

$$C = \begin{bmatrix} F \\ K \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \end{bmatrix}$$

plain text $P = K^{-1}C \bmod 26$.

$$= \begin{bmatrix} 2 & 25 \\ 25 & 18 \end{bmatrix}\begin{bmatrix} 5 \\ 10 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 260 \\ 305 \end{bmatrix} \bmod 26 = \begin{bmatrix} 0 \\ 19 \end{bmatrix} = \begin{bmatrix} A \\ T \end{bmatrix}$$

Similarly for MF = $\begin{bmatrix} T \\ A \end{bmatrix}$

IO = $\begin{bmatrix} C \\ K \end{bmatrix}$.

One-time pad

One time pad is a type of Vigenere cipher which includes the following features -

→ It is an unbreakable cipher

→ The key is exactly same as the length of message which is encrypted

→ The key is made of random symbols

→ As the name suggests, key is used one time only and never used again for any other message to be encrypted.

Due to this, encrypted message will be vulnerable to attack for a cryptanalyst. The key used for one time only and never used again for any other message to be encrypted pad, cipher is called pad, as it is printed on pads of paper.

playfair cipher Decryption Algorithm:

The Algorithm consists of 2 steps:

1) Generate the key square (5×5) at the receiver's end:

→ The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique, and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.

→ The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

2) Algorithm to decrypt the ciphertext:

The ciphertext is split into pairs of two letters (digraphs).

Note: The ciphertext always have even number of characters

Example

Cipher text: "gatlmzclrqtx"

After split: ga tl mz cl rq tx

Rules for Decryption

→ If both the letters are in the same column take the letter above each one (going back to the bottom if at the top)

→ If both the letters are in the same row: Take the letter to the left of each one (going back to the rightmost if at the leftmost position)

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

3) If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle

Diagraph: cl
Decrypted text: me
c → m
l → e

Diagraph: tl
Decrypted text: st
t → s
l → t

Diagraph: rq
Decrypted text: nt
r → n
q → t

Transposition techniques.

→ In Transposition techniques there is no replacement of characters

→ we will rearrange the characters position i.e., we will apply some sort of permutation on the plaintext letters.

i) Rail Fence technique

In this, the plaintext is written down as a sequence of diagraph and then read off as a sequence of rows.

Eg: "all the best for exams" → plain text

To encrypt this with a rail fence of depth 2 we write the following

```
a l h b s f r x m s
 l t e e t o e a a
```

Encrypted msg is: Alhbsfrxmsltecteoeaas.

Note: used for short messages.
easy to break by the attackers

Row Transposition Cipher

We write the message in a rectangle row by row and read the message off column by column, but permute the order of columns

Key - Integer value (unique digits from 0 to 7)

eg: 4 5 3 1 2 ← CRYPTO
1 4 6 3 5 2

eg: 4321

Eg: plain → attack postponed until two am

Key → 4 3 1 2 5 6 7

a t t a c k p
o s t p o n e
d u n t i l t
w o a m x z y

dummy bit.

Cipher text: ttna aptm co i x kn l2 pety aodw
tsuo

Explanation: To encrypt start with the column labelled as 1 ie, in our ex 3

problem → can easily be understood by the attacker
used for short msg only

It can be made more secure by performing more than 1 stage of transposition so, the result will be a more complex permutation

4	3	1	2	5	6	7
t	t	n	a	a	p	t
m	t	s	u	o	a	o
d	w	e	o	i	x	k
n	l	y	p	e	t	z

o/p: nsey auop TTwl Tmdn aoie Part tolz

Double Transposition

Columnar transposition / Row transposition cipher applied twice.

→ The key in case 2 can be same / different

→ This technique was used in world war - I by German military and also in world war II

Note: eg: keyword / key → STRIPE

It will be used as → 5 6 4 2 3 1 } decided by the alphabetical order of letters in the key

eg: Key: ZEBRA
5 3 2 4 1

Keyless Transposition Ciphers

Simple transposition ciphers, which were used in the past are keyless. There are two methods for permutation of characters. In the first method, the text is written into a table column by column and then transmitted row by row. In the second method, the text is written into the table row by row and then transmitted by columns by column.

Eg: Meet me at the park

m e e t
m e a t
t h e p
a r k

mematteakethpr

m e e t
m e a t
t h e p
a r k

mmta eehr aaekttp.

Keyed Transposition ciphers

The keyless ciphers permute the characters by using writing plaintext in one way (row by row) and reading it in another way (col by col). Another method is to divide the plaintext into groups of predetermined size called blocks and then use a key to permute the characters in each block seperately.

enemy attacks tonight to Bob.

key used for Encryption and Decryption } → permutation key.

Assume ↓ Encryption

	3	1	4	5	2
	1	2	3	4	5

↑ Decryption

3	1	4	5	2
e	n	e	m	y
a	t	t	a	c
k	s	t	o	n
i	g	h	t	t
o	b	o	b	x

Cipher: ntsgb
etthomaotb
ycntxeakio.

Modern Symmetric Key Encryption

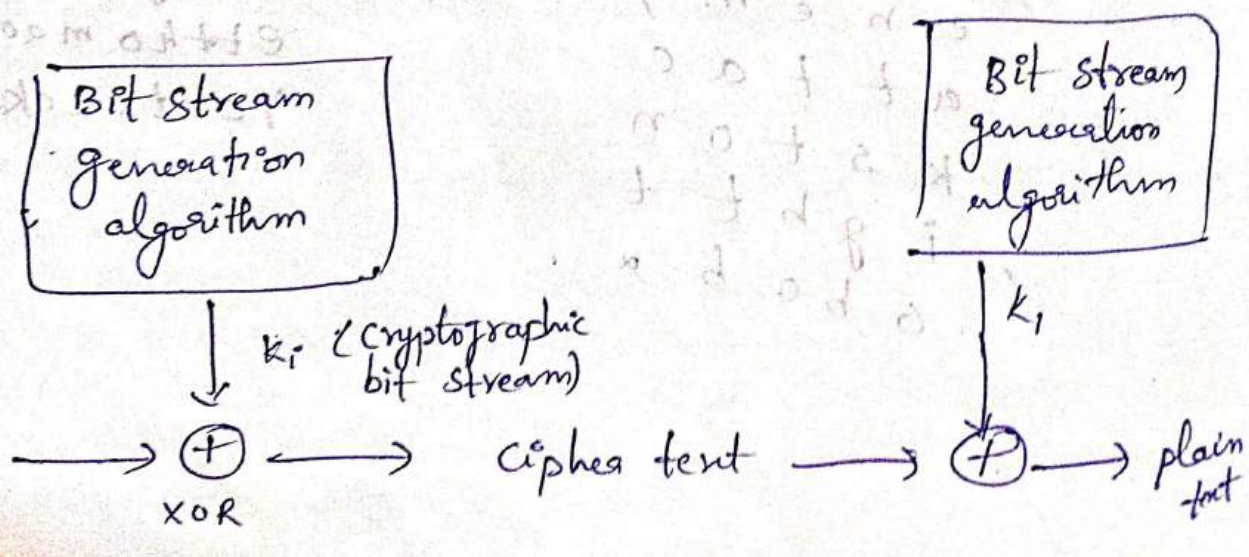
Digital data is represented in strings of binary digits unlike alphabets. Modern cryptosystems need to process this binary strings to convert it into another binary strings are processed, a symmetric Encryption schemes can be classified into

* Block ciphers
* Stream ciphers

Stream cipher

It is the one that encrypts a digital data stream one bit or 1 byte at a time

It is a symmetric key cipher (i.e., 1 key for encrypt + decrypt)



eg
```
1 0 1 1 0 1 1 0   — Message (Sender)
0 1 0 1 0 1 0 1   — Key.
-----------------
1 1 1 0 0 0 1 1   — Cipher
-----------------
```

To decrypt

```
1 1 1 0 0 0 1 1   — Cipher
0 1 0 1 0 1 0 1   — Key.
-----------------
1 0 1 1 0 1 1 0
-----------------
```
XOR

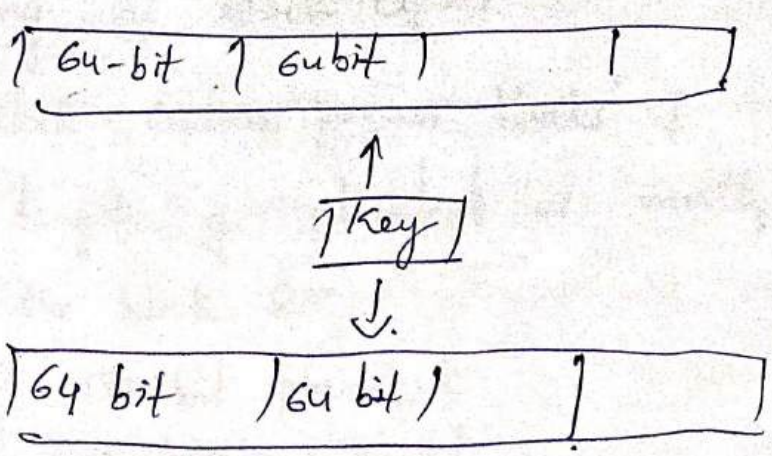
Block cipher.

In this, a block of plain text is treated as a whole and used to produce the cipher text of equal length.

Typically a block size of 64 and 128 bits is used

Symmetric key cipher (1 key used only)

key will be applied on each block.



plain text in blocks

cipher text will also be in blocks.

Eg of Block Cipher

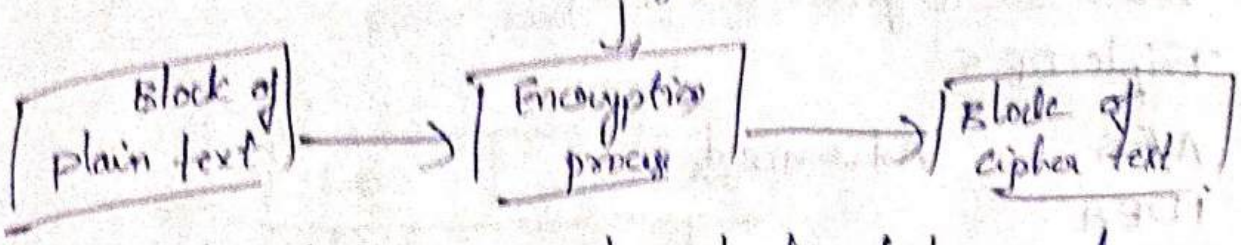
→ DES (64 bit block size) cipher and plain text

Differences between Block cipher & Stream Cipher

Block cipher	Stream cipher
1) Block Cipher converts the plaintext into cipher text by taking plain text's block at a time	1) Stream Cipher converts the plain text into cipher text by taking 1 byte of plain text at a time
2) Block Cipher uses either 64 bits or more than 64 bits	2) while stream cipher uses 8 bits
3) The Complexity of block Cipher is simple	3) while stream cipher is more Complex
4) In block Cipher, reverse encrypted text is hard	4) while in stream Cipher, reverse encrypted text is easy

Block cipher

Basic scheme of a block cipher



→ Takes block of plaintext bits and generates a block of ciphertext bits generally of same size.
→ Size of the block is fixed.
→ Block size does not directly affect to the strength of encryption scheme.
→ Strength of cipher depends up on the key length

Block size

The following aspects should be in mind while selecting a size of a block

→ Avoid very small block size
→ Do not have very large block size
→ Multiples of 8 bit.

padding in Block Cipher

Block ciphers process blocks of fixed sizes (say 64 bit). The length of plaintext is mostly not a multiple of the block size

Eg: 150 bit plain-text
2 blocks - 64 bit
1 block - 22 bits

this needs to be padded to make it 64 bit

The process of adding bits to the last block is padding

Block cipher scheme

Most popular and prominent block ciphers are : DES – Digital Encryption
Triple DES
AES – Advanced
IDEA
Twofish Blowfish
Serpent

Feistel Block Cipher.

It is not a specific scheme of Block Cipher It is design model from which many different Block ciphers are derived

→ In Feistel Cipher, the plain text is divided in 2 equal halves

→ The 2 halves of the data pass through n rounds of processing and then combine to produce the ciphered block.

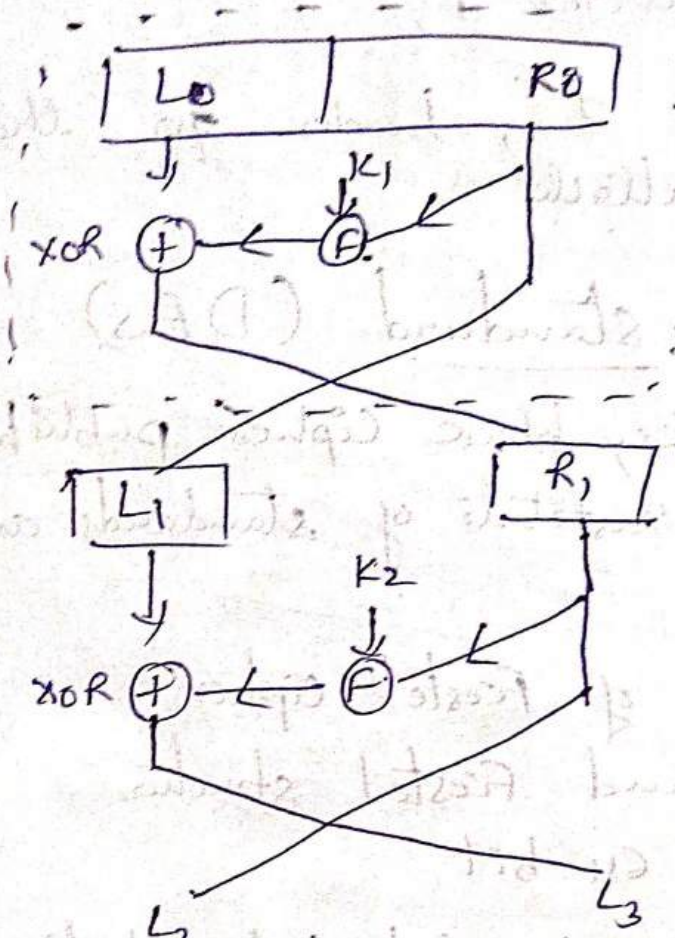
→ one the right half we apply a function and in the fn we will need a subkey generated from the master key.

The output of this is XORed with the left half and then their o/p will be swapped.

This is one single round.

we will have [n rounds] → depends on Algorithm
All the rounds will have same structure

plain text divided in 2 equal halves.



Ciphertext block.

1) Block size – Larger block size, more security
2) Key size → Larger key size means more security but may decrease the speed of encryption/decryption
3) No. of rounds → More rounds, more secure

4) Subkey generation algorithm.

More complex algorithm, algo harder for attacker to steal data

5) function / Round function F

more complex fn, harder for the cryptanalyst to attack

Digital Encryption standard (DES)

→ Symmetric-key block ciphers published by the National institute of standards and Technology

— implementation of Feistel cipher.
— uses 16 round Feistel structure
— Block size 64 bit
— Key length 64-bit, but effective key length is 56 bit since 8 of the 64 bits are not used by the encryption algorithm (function as check bits only).

D.E.S :- Data Encryption Standard ...

(i) block cipher

(ii) Symmetric cipher (same key for encryption + decryption)

(iii) 64 bit plain text block.
It encrypts the data in blocks of size 64 bits each.

(iv) 16 rounds . each round is a fiestel round.

Steps

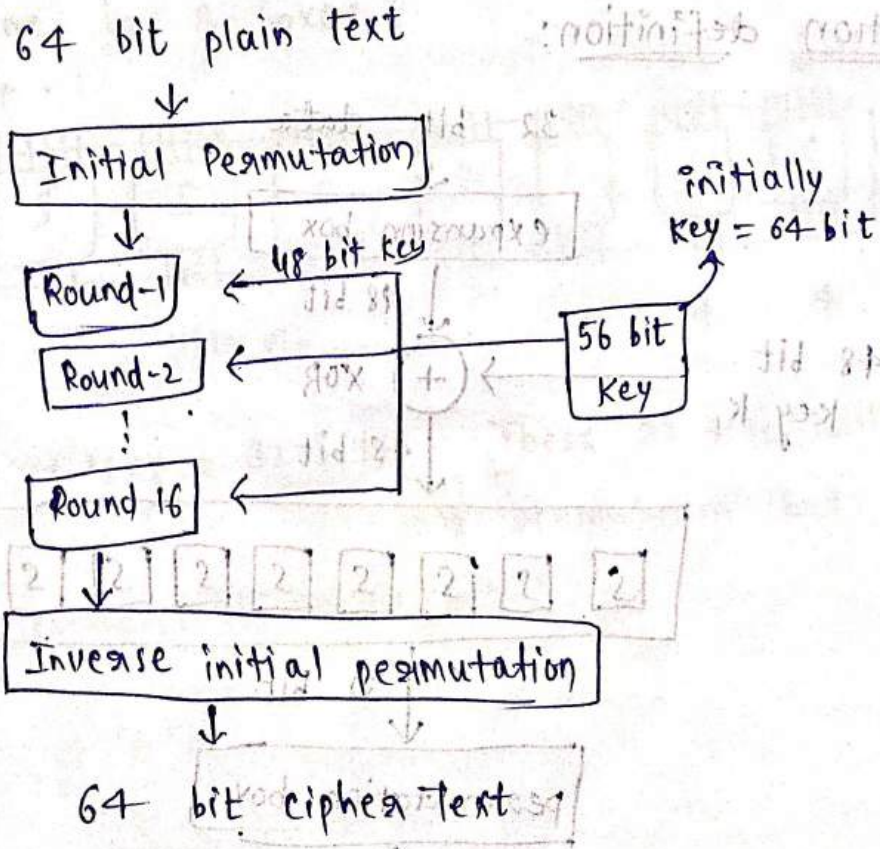
(i) Initial permutation

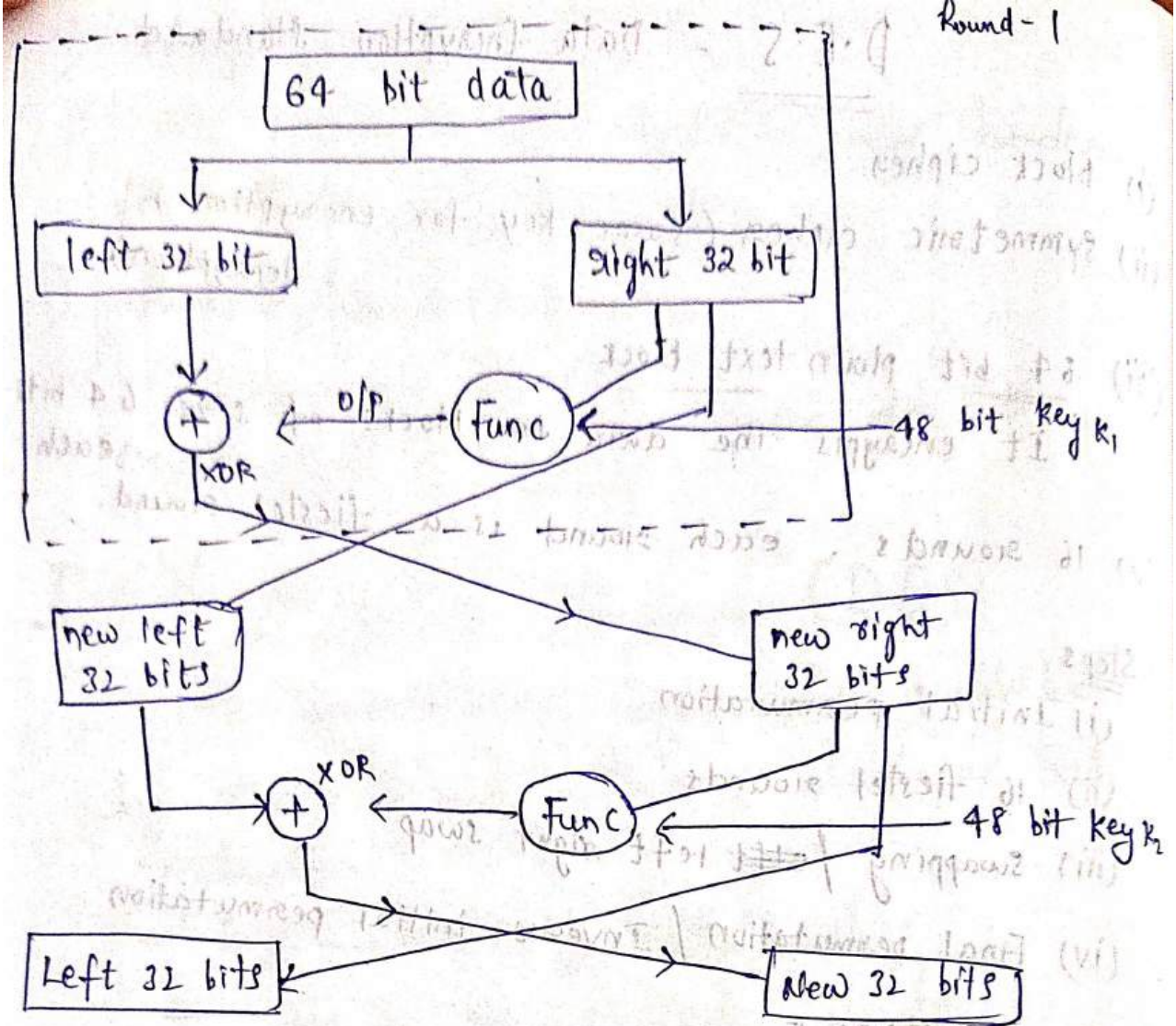
(ii) 16 fiestel rounds

(iii) Swapping / ~~eft~~ left right swap

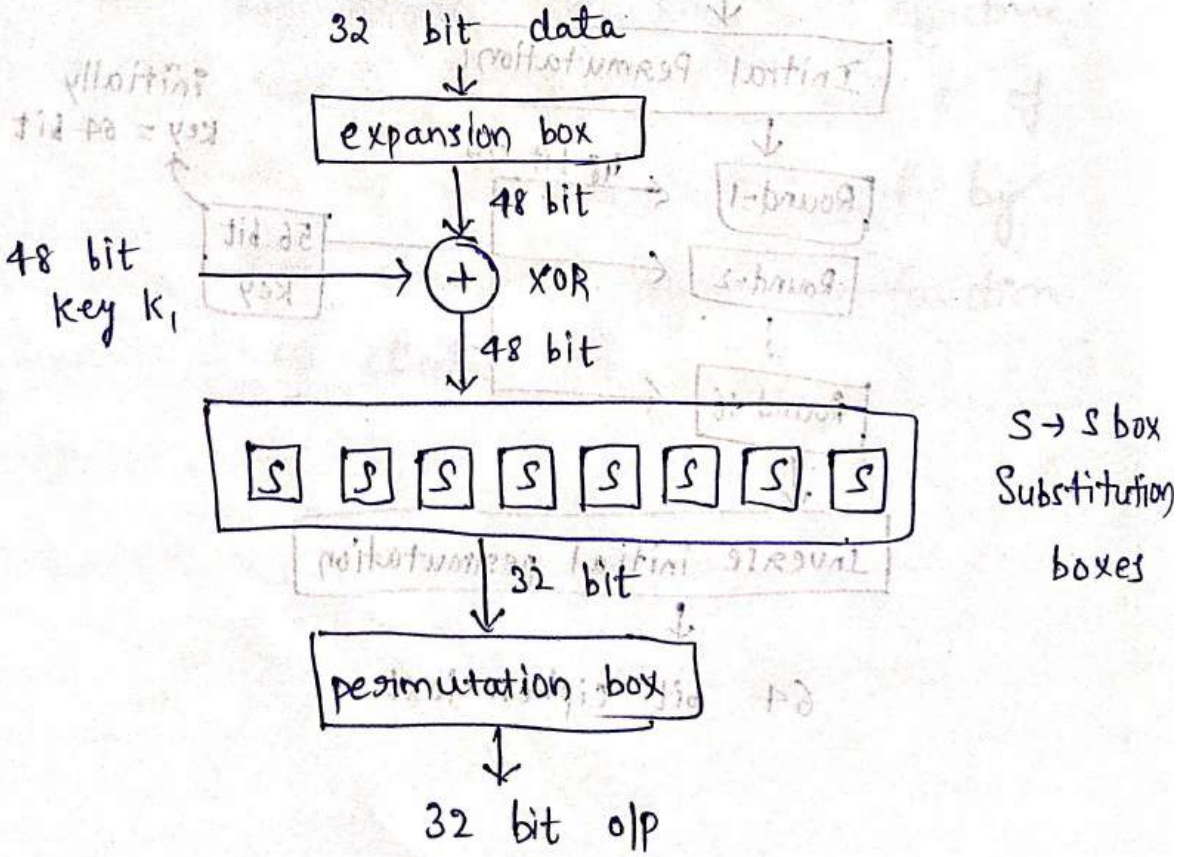
(iv) Final permutation / Inverse initial permutation

Basic Structure :-



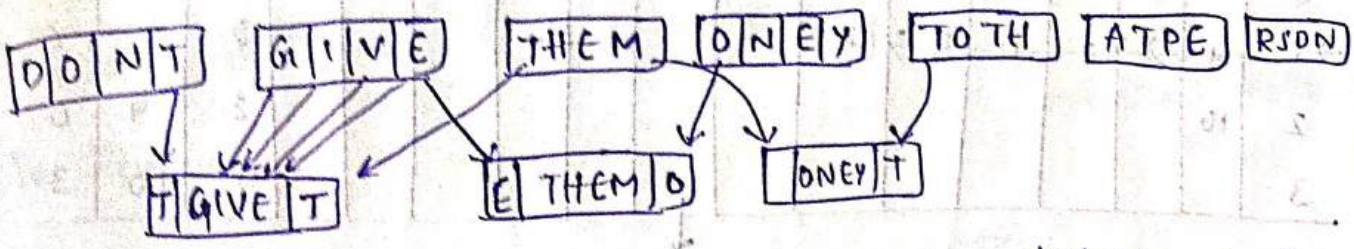


Function definition:



*What happens in expansion box?

32 bit data will be → 1's and 0's form but for explanation let us consider a text



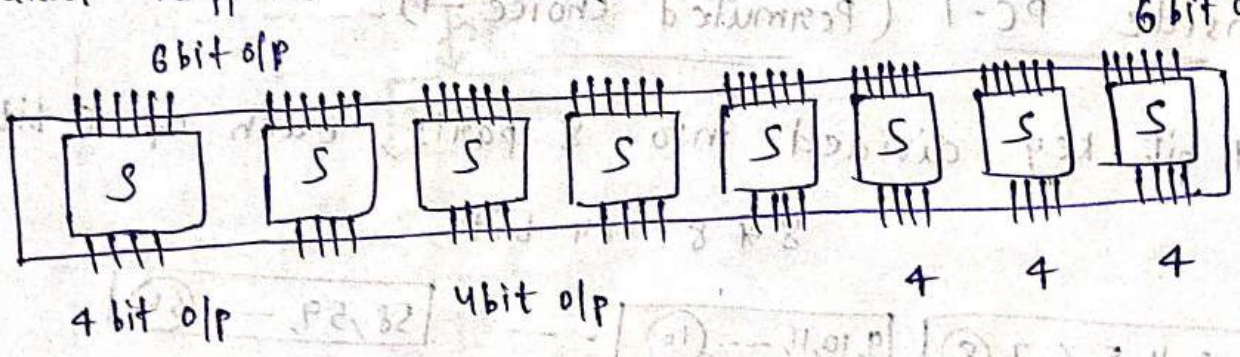
So here every 4 bit block is converted to a 6 bit block.

There were 8 blocks of 4 bit each = 32 bit

Now, there are 8 blocks of 6 bit each = 48 bit

Now these 48 bits XOR with 48 bit key and given sent to S-boxes.

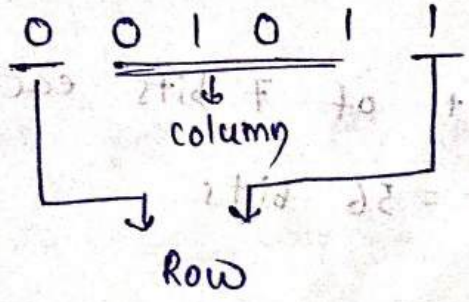
*What happens in S-boxes?



o/p → 4x8 = 32 bits. These 32 bits will go into permutation box.

*How 6 bit converted to 4 bit?

→ eg:



01 → 1

0101 → 5

	0	1	2	3	4	(5)	6	7	8	9	10	11	12	13	14	15
0	3	5	7	0	1	4	.	.	.	.	.	.	.	1	4	5
→1	4	2	1	0	9	(7)								2	3	6
2	10													3	4	0
3														11	10	3

10 → 1010 numbers will be filled.

Each s box will have a def table.

* How 16 subkeys are generated?

→ actually, we have 64 bit key which go as a i/p to PC-1 (permuted choice -1) and we get o/p as 56 bit key.

* Inside PC-1 (Permuted choice -1):

64 bit key divided into 8 parts, each of 8 bit.

8 * 8 = 64 bit

[1 2 3 4 5 6 7 (8)] [9, 10, 11, ---, (16)] ---- [58, 59, ---, (64)]

1st part 2nd part/block 8th part

From each part, last bit → discarded

i.e, bit → 8, 16, 24, 32, --- 64 discarded

Hence, we have 8 parts of 7 bits each

= 8 x 7 = 56 bits

→ O/p of PC-1 is 56 bits which is then divided into 2 parts of 28 bits each → C_0, D_0

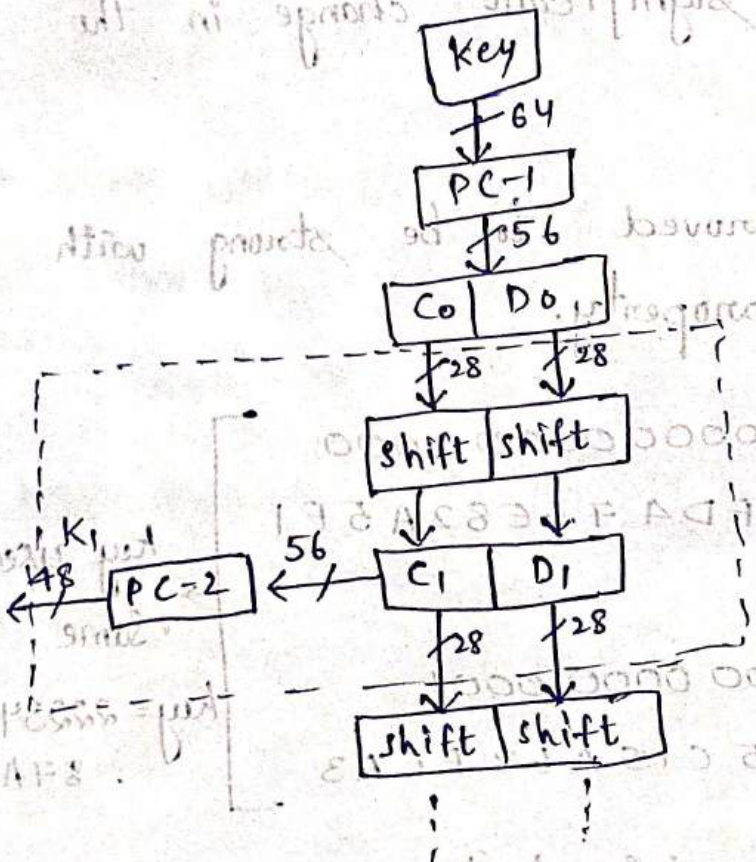
Now these bits are shifted with left shift in each round.

in Rounds i = 1, 2, 9, 16 → 1 shift i.e rotated left by 1 bit

in other rounds,

two halves rotated left by 2 bits.

→ After shifting, we get (C_1, D_1) which goes as i/p to PC-2



Inside PC-2 56 bit → 48 bit using a predefined table

Then we get our 1st key for Round 1

In C_1 → 28 bits → (1-28)

D_1 → 28 bits → (29-56)

*How 48 bits were selected from 56 bits?

Left half C_1 (9, 8, 22, 25 position bits are missing)
ie 24 left

Right half D_1 (35, 38, 43, 54 position bits are missing)
i.e removed
i.e 24 left

~~10 → 12~~
~~1 → 6~~

Properties

(i) Avalanche effect:

It means a small change in plain text (or key) should create a significant change in the cipher text

→ DES has been proved to be strong with regard to this property.

eg: Plain → 0000 0000 0000 0000
Cipher → 4789 FD47 6E82 A5F1

Plain → 0000 0000 0000 0001
Cipher → 0A4E D5C1 5A63 FEA3

→ Key used is same.
Key = 22234512987ABB23

→ Although, the two plain texts differ only in 1 bit, cipher text block differs a lot (or) significantly.

(ii) completness effect:

It means that, each bit of the cipher text needs to depend on many bits on the plain text.

→ The confusion and diffusion produced by D-boxes and S-boxes in DES, show a very strong completeness effect.

DES Weakness

(i) Key size

→ Critics believe that the most serious weakness of DES is its key size of 56 bits.

→ With today's technology (like parallel processing and very powerful processors) it can easily be cracked ↓ 2^{56} keys (brute force attack)

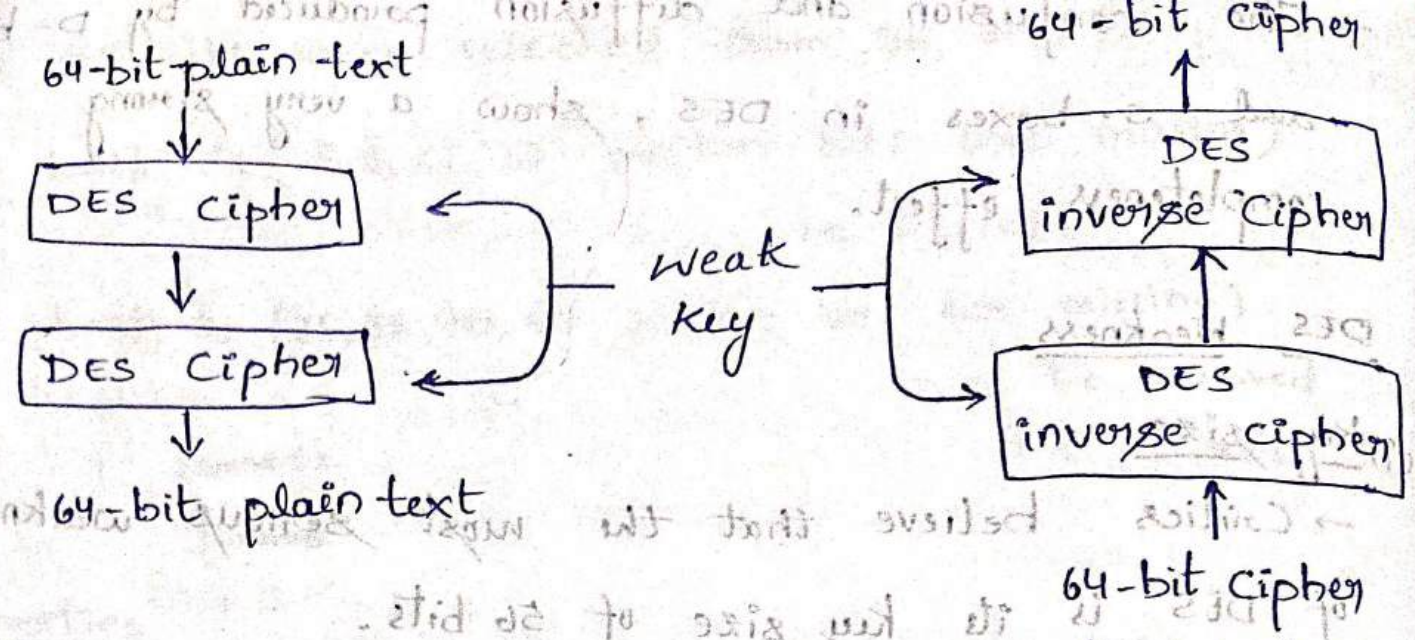
→ ∴ we use triple DES (3DES) with two keys (112 bits) or triple DES with 3 keys (168 bits)

(ii) Weak Keys → Four out of 2^{56} keys are called weak keys.

→ A weak key is the one which after the parity drop operation consists of all 0's, all 1's or half 0's and 1's.

The disadvantage of using a weak key is:

If we encrypt a block with a weak key and subsequently encrypt the results with the same weak key, we get the original block.



→ The process creates the same original block if we decrypt the block twice

→ So, if after 2 decryptions, if the result is the same then, attacker is successful.

(iii) Semi Weak Keys: Six key pairs are called semi-weak keys

→ A semi-weak keys creates only two different round keys, and thus each of them is repeated 8 times

(iv) Possible weak keys:

→ There are 48 keys that are called possible weak keys.

→ A possible weak key is a key that creates only 4 distinct round keys, in other words the 16 round keys are divided into 4 groups and each group is made of 4 equal keys.

(v) Key clustering

Means 2 or more dif keys can create the same cipher text from the plain text.

Weakness in Cipher Design

(i) Two specifically chosen i/p's to s-box array can create the same o/p

MULTIPLE DES

1. Double DES

→ uses 2 dif keys: (56 + 56) = 112 bit key

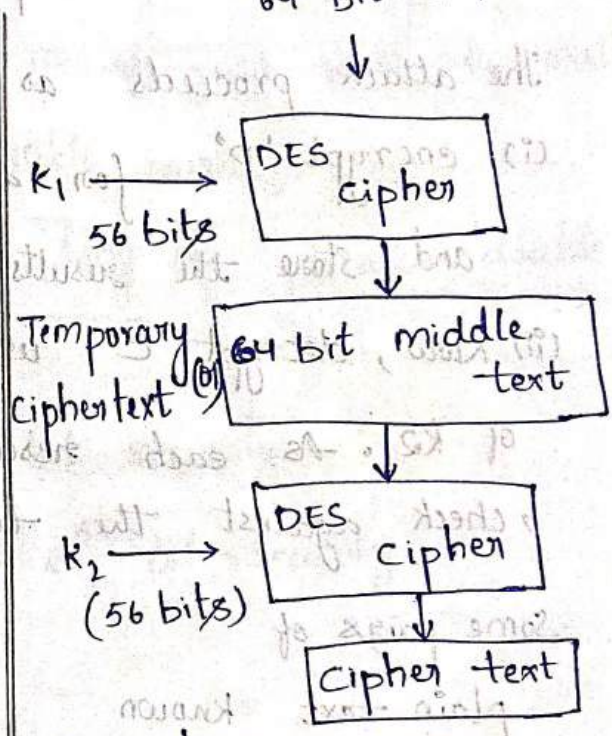
→ Double encryption occurs as follows:

$P \longrightarrow E(K_1, P)$

$\downarrow$

$E(K_2, E(K_1, P)) = \text{Cipher}$

→ for decryption, 1st decrypted using key K_2 which produces single encrypted cipher text



Encryption block diagram of 2DES

→ This 64 bit middle text / temp. cipher text is then decrypted using the key K_1 to get plain text

$$\text{Plain} = D(K_1, D(K_2, C))$$

Drawbacks of Double DES

i) Meet-in-the-middle attack (MIM attack)

This attack involves encryption from 1 end and decryption from the other end and then "matching the results in the middle" and hence the name.

→ This attack requires knowing some plaintext / ciphertext pairs

Let us assume plaintext = P ciphertext = C

The attack proceeds as follows:

(i) encrypt 'P' for all 2^{56} possible values of K_1 and store the results in a table and sort it.

(ii) Now, decrypt 'C' using all 2^{56} possible values of K2. As each result (i.e., decryption) is produced, check against the table for match.

Some pairs of

plain text known & Cipher text known

encrypt pairs for all 2^{56} possible values of K1

decrypt pairs of all 2^{56} possible values of K2

plain	cipher	middle
ABCD	XYMP	
X$#T	MXMT	
—	—	
—	—	

Cipher	middle Cipher
—	—
—	—
—	—

no. of rows in table = no. of possible secret keys.

→ we will compare the values of 2nd table with the values of the 1st table computed earlier.

$$\text{Decrypt}(K_2, C) = \text{Encrypt}(K_1, P)$$

∴ (K_1, K_2) is the key pair used.

(iii) when there is a match, we have located a possibly correct pair of keys.

Note → Now, more than 1 pair of keys may result in a match but, these no. of pairs will be small, we should try each possible pair of keys.

So, it takes twice as long to break double DES using brute force.

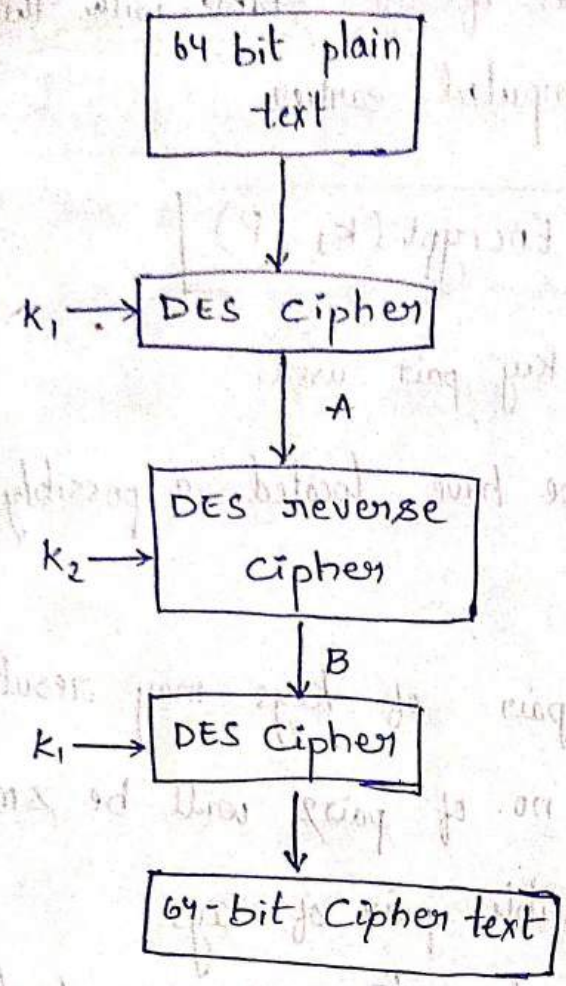
because DES has 2^{56} bit security,

double DES has $2 \times 2^{56} = 2^{57}$ bit security.

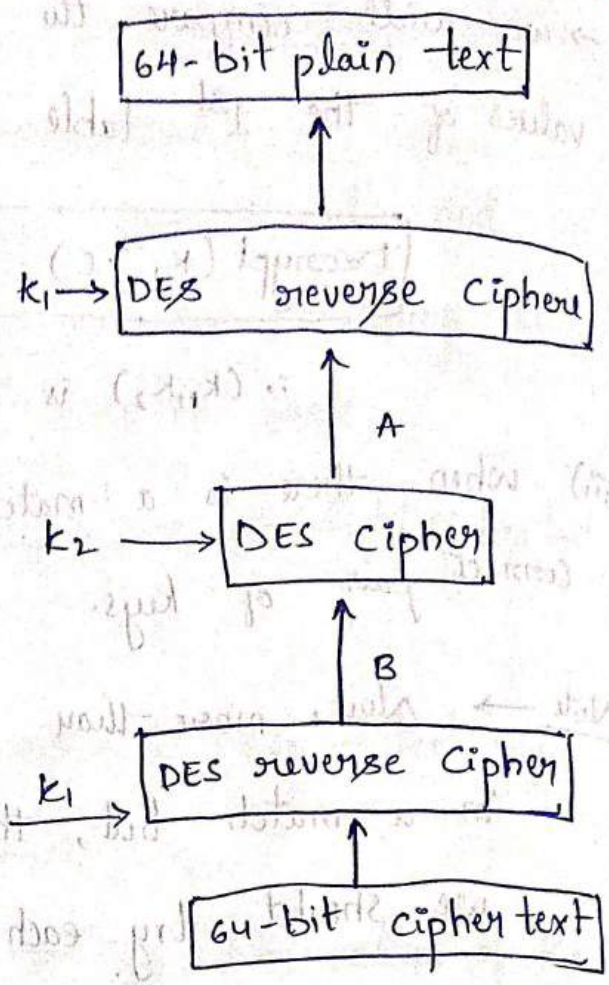
2. Triple DES (3DES)

→ 2 or 3 keys are used

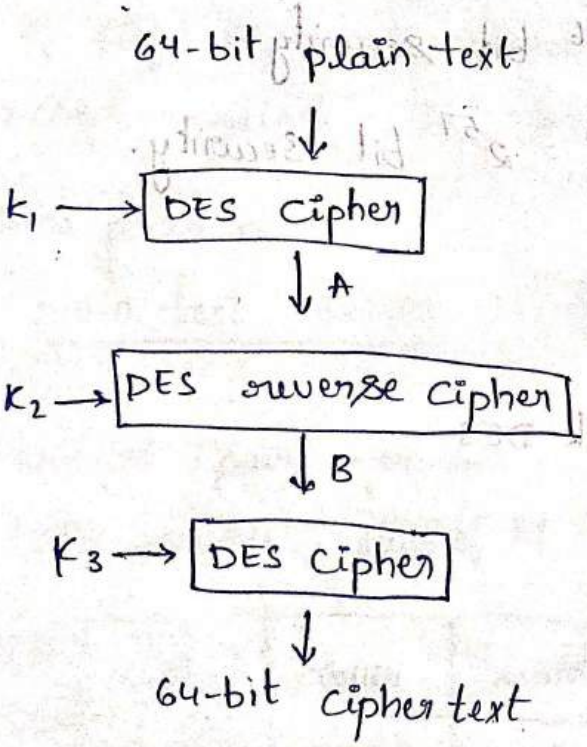
→ much stronger than double DES



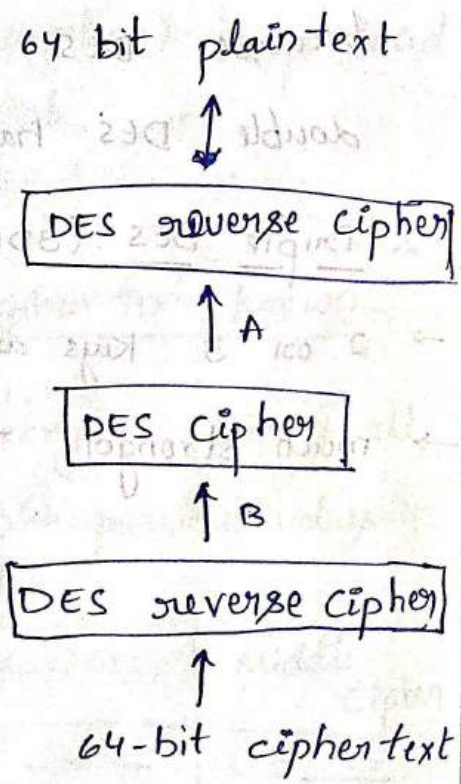
encryption using 2keys



decryption using 2keys



encryption using 3 keys

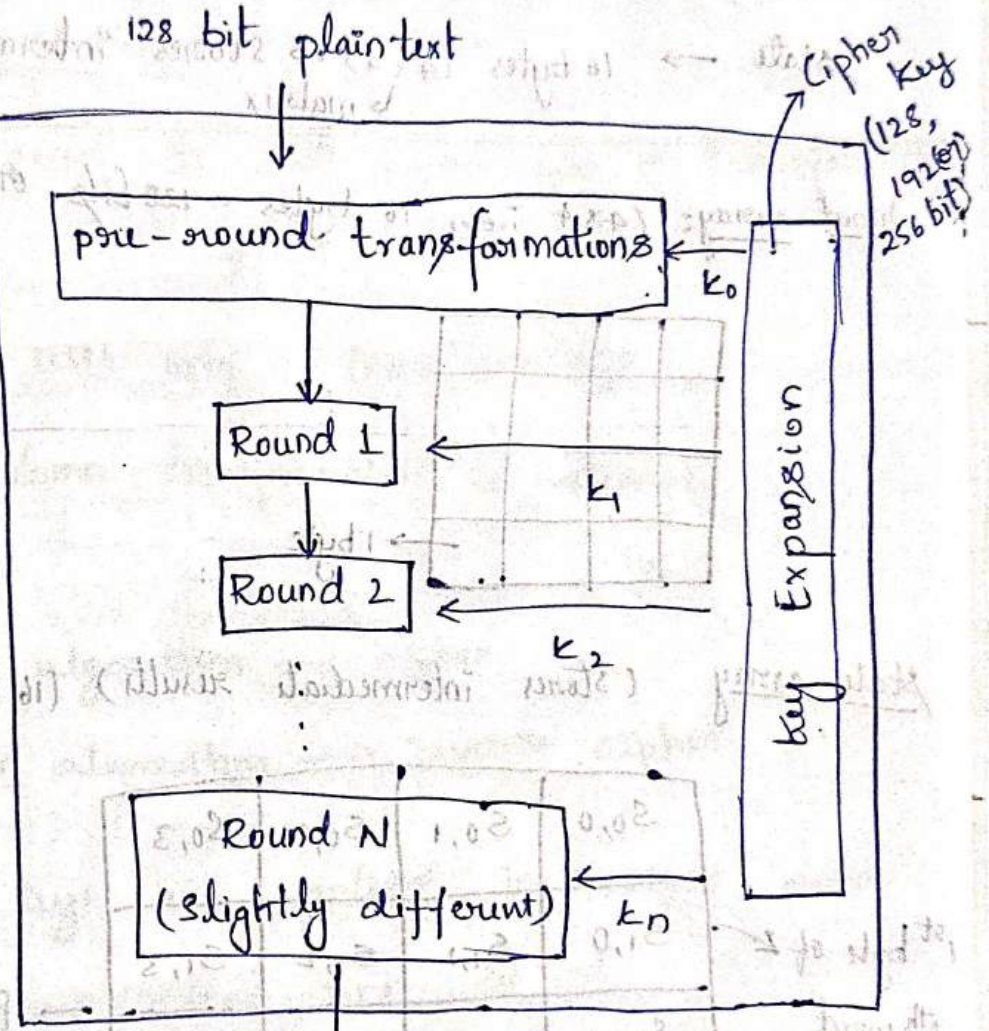


decryption using 3Keys

Advanced Encryption Standard (AES)

- Symmetric key block cipher
  (i.e., same key used for encryption & decryption)
- established in 2001 by the U.S NSIT
- Fixed block size = 128 bits i.e., 16 bytes = 4 words
  (∵ 1 word = 32 bits)

General design of AES Encryption



Rounds	no. of bits in key	
10	128	→ AES-128 version
12	192	→ AES-192-version
14	256	→ AES-256-version

→ no. of keys generated by key expansion algorithm

= (no. of rounds + 1)

Note:

bit → 0 (or) 1

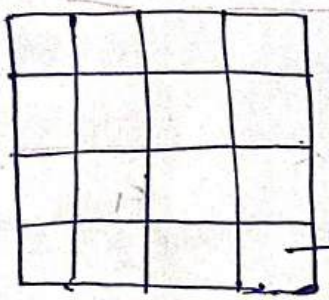
1 byte → group of 8 bits

1 word → 4 bytes = 32 bits

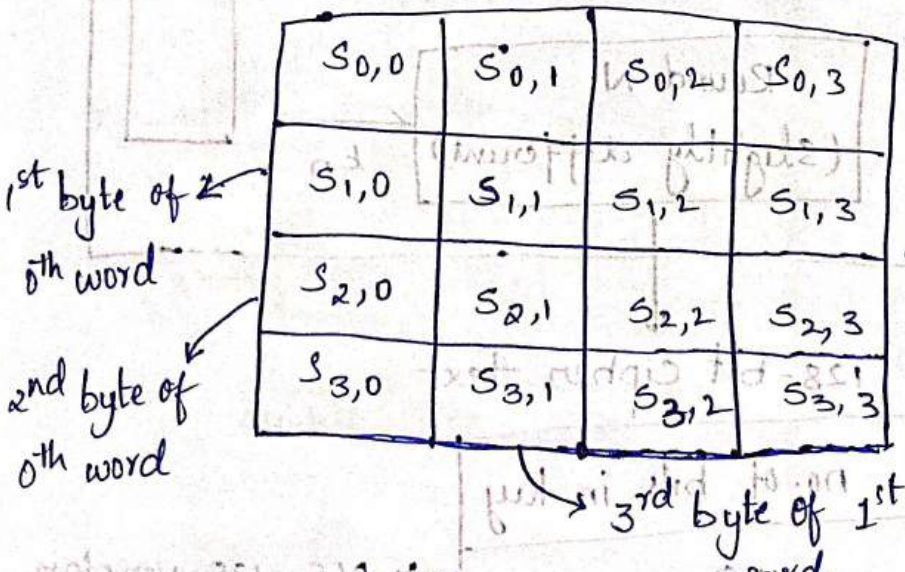
Block size = 128 bit data

State → 16 bytes (4×4) → stores intermediate results
↳ matrix

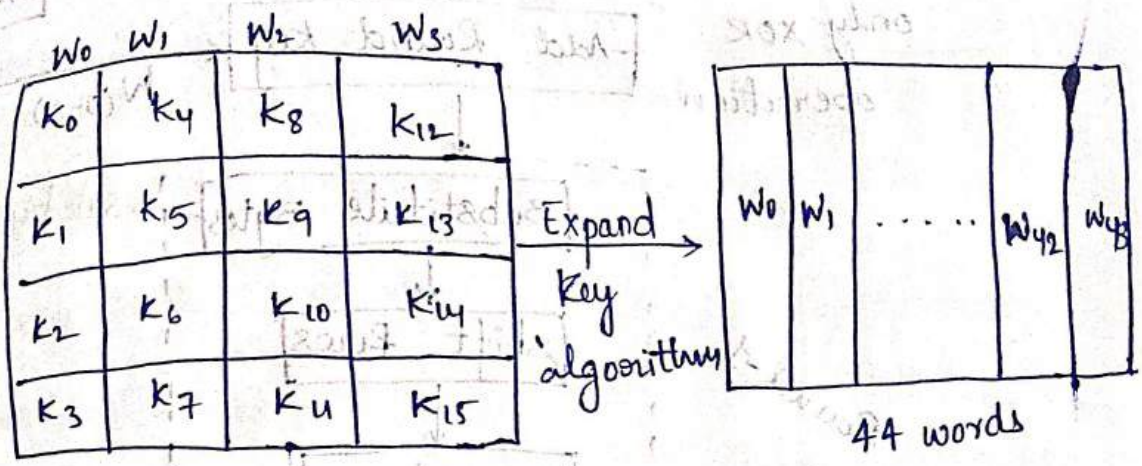
Input array: (4×4 i.e., 16 bytes = 128 bits or 4 words)



State array (stores intermediate results) (16 bytes)



Key → 128 bits = 4 words



Structure of Each Round:

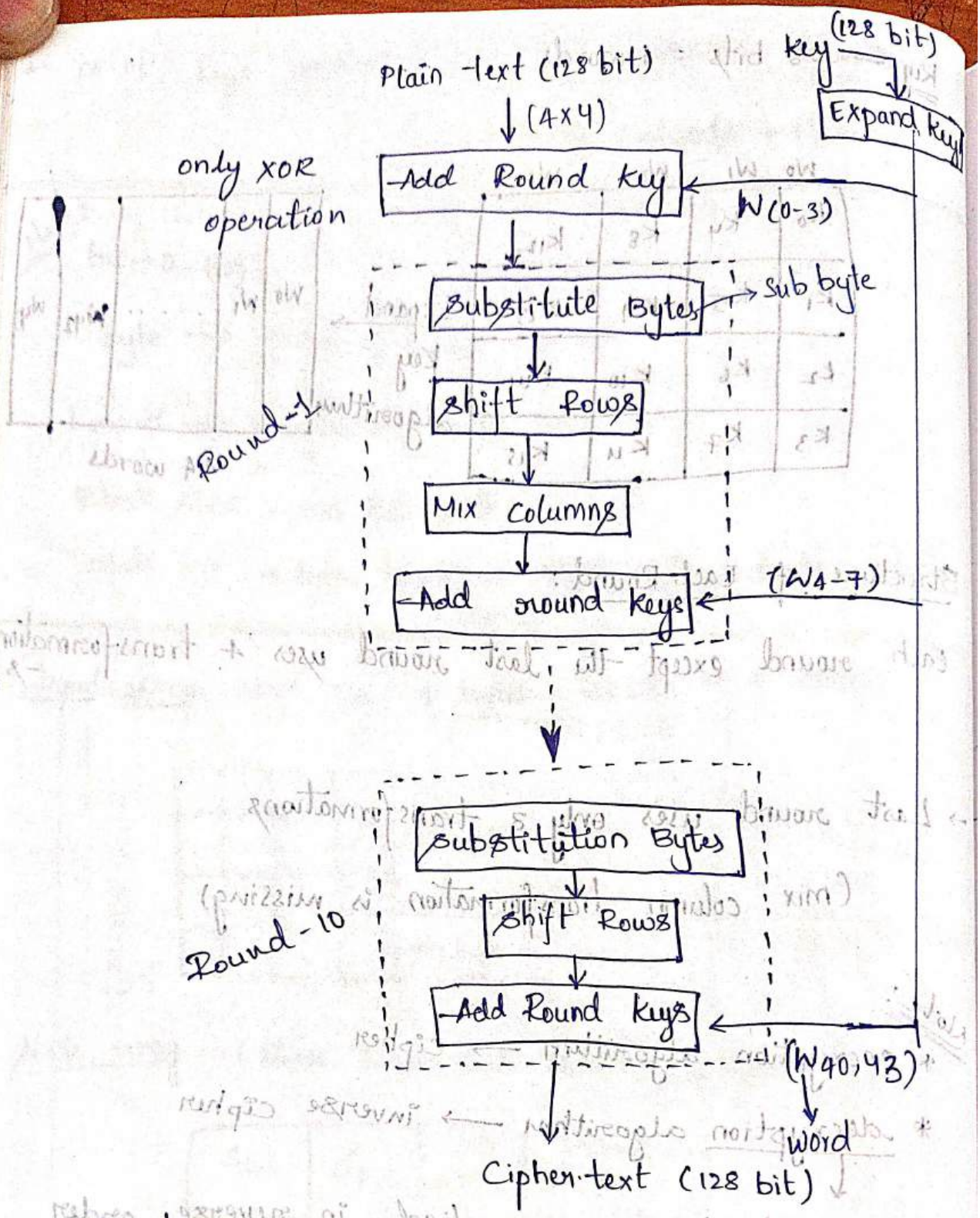
Each round except the last round uses 4 transformations

→ Last round uses only 3 transformations

(mix column transformation is missing)

Note:

* encryption algorithm → cipher
* decryption algorithm → inverse cipher
  ↓
  round keys are applied in reverse order
* Much stronger than DES



Transformations

4 types substitution, permutation, mixing, and key-adding.

1) substitution:

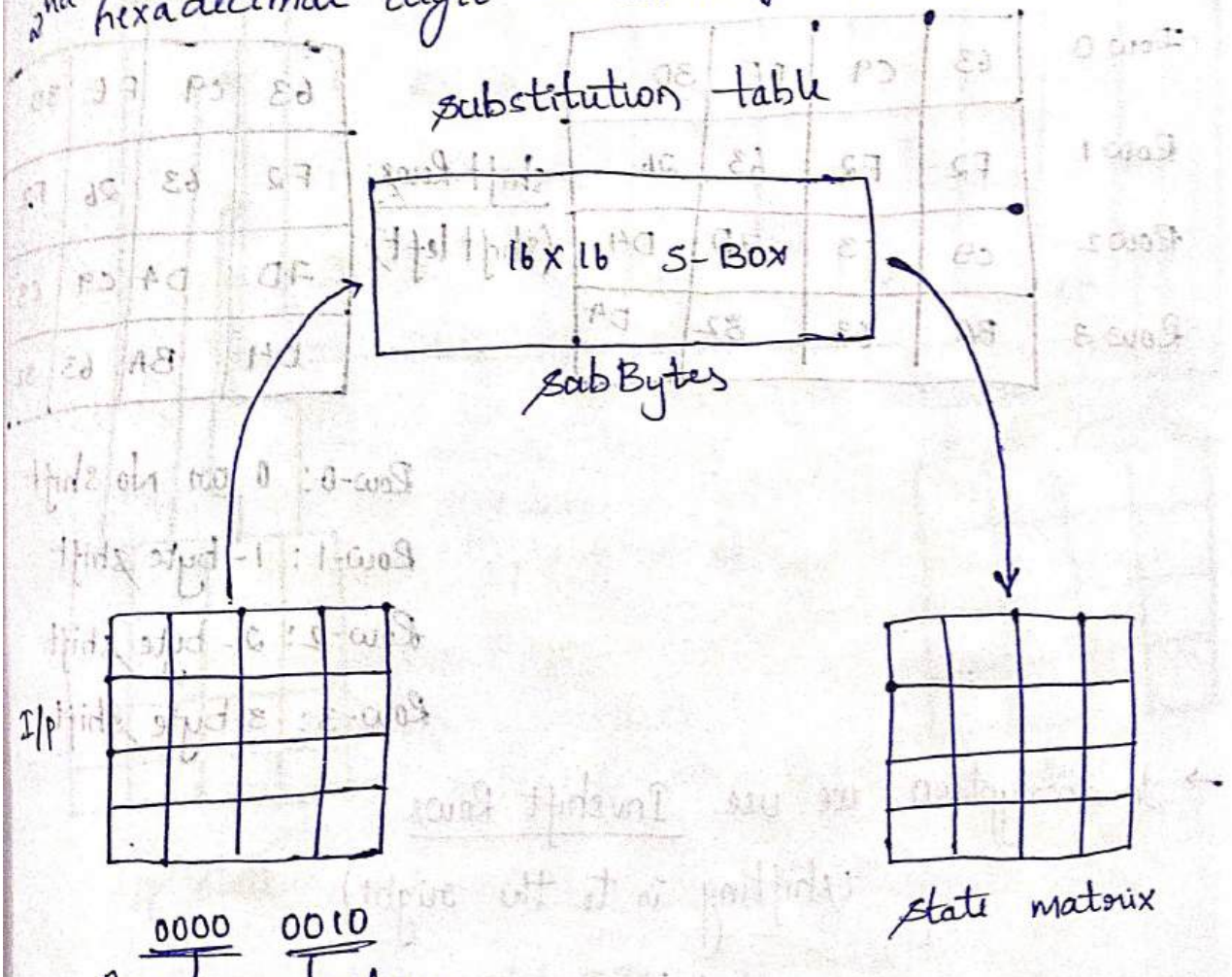
AES like DES uses substitution, but mechanism is different. substitution is done for each byte

- Only one table is used for transformation of bytes which means that if 2 bytes are same, the transformation is also same.

Sub Bytes :- at encryption side

we interpret the byte as 2 hexadecimal digits.

1st hexadecimal digit → row
2nd hexadecimal digit → column } of the substitution table



↳ Transformation is done one byte at a time

2. Permutation:

In this we permute / shift the bytes.

→ In DES, permutation was done at bit level

In AES, permutation is done at byte level

Shift Rows

→ Shifting is done to the left

→ no. of shifts depends on the rows of the state matrix

Row 0	63	C9	FE	30
Row 1	F2	F2	63	26
Row 2	C9	C3	7D	D4
Row 3	BA	63	82	D4

shift Rows (shift left) →

63	C9	FE	30
F2	63	26	F2
7D	D4	C9	C3
D4	BA	63	82

Row-0: 0 or No shift

Row-1: 1-byte shift

Row-2: 2-byte shift

Row-3: 3-byte shift

→ In decryption, we use Invshift Rows (shifting is to the right)

no. of shifts is same

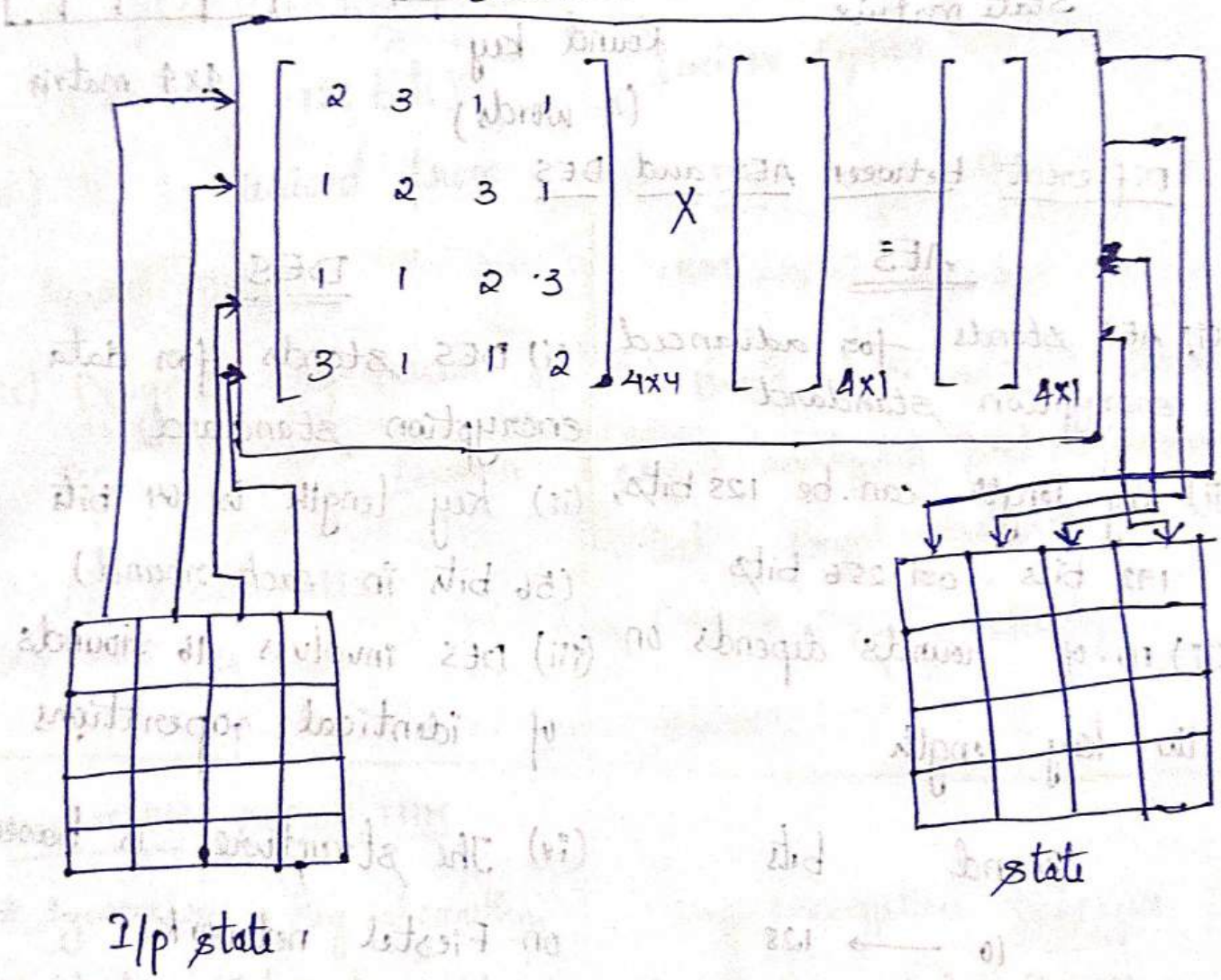
Note:

Shift Rows & Invshift Rows } transformations are inverse of each other.

3. Mixing

Mix columns - for encryption

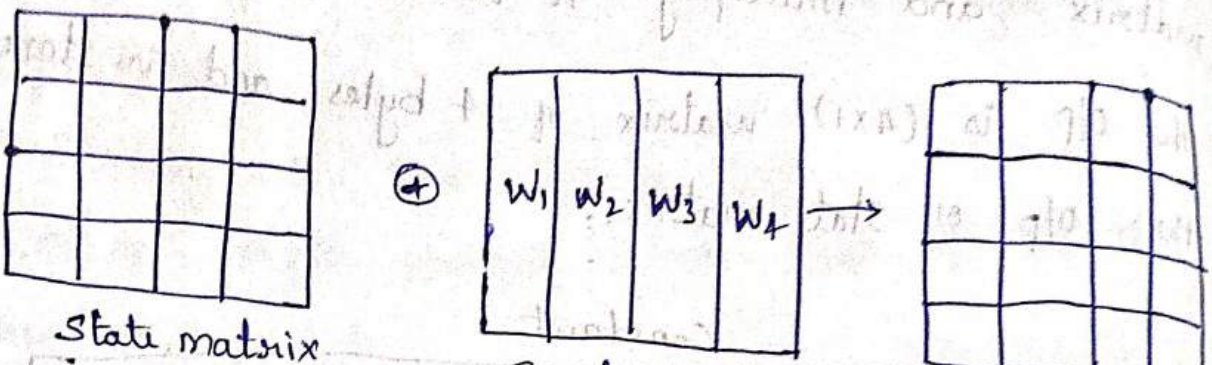
Take each word / column i.e., 4 bytes or 4×1 matrix and multiply it with the constant matrix. The o/p is (4×1) matrix of 4 bytes and is stored in the o/p or state matrix.



MIX COLUMN TRANSFORMATION

4. Key -Adding

-Add Round key - also proceeds 1 columns at a time



State matrix

Round key (4 words)

4x4 matrix

Different between AES and DES

AES	DES
(i) AES stands for advanced encryption standard	(i) DES stands for data encryption standard
(ii) Key length can be 128 bits, 192 bits or 256 bits	(ii) key length is 64 bits (56 bits in each round)
(iii) no. of rounds depends on the key length Round bits 10 ⟶ 128 12 ⟶ 192 14 ⟶ 256	(iii) DES involves 16 rounds of identical operations
(iv) The structure is based on the substitution-permutation networks	(iv) The structure is based on Fiestel network.
	(v) It is less secure. It can be broken down (i.e., it is weak) 3DES more secure than DES

(v) AES is more secure than DES and is the de-facto world standard

(vi) Rounds in AES are: byte substitution, shift Row, Mix column and Key addition

(vii) It can encrypt 128 bits of plaintext (i.e., block size is 128 bits)

(viii) It is derived from Square cipher

(ix) Designed by vincent Rijmen & Joan Daemen

(x) No known attack

(xi) AES is faster

(vi) Rounds in DES are: Expansion, XOR operation with round key, substitution and permutation

(vii) It can encrypt 64 bits of plain text

(viii) It is derived from Lucifer Cipher

(ix) DES was designed by IBM.

(x) Brute force attack, Linear crypt-analysis and differential crypt-analysis

(xi) It is comparatively slower

BLOWFISH ALGORITHM

* Symmetric key algorithm || encryption-technique
* block cipher (64 bit) algorithm || designed by Bruce Schneier (1993)
* It is an alternative to DES encryption technique & IDEA algorithm

Block size / plain-text in 1 block → 64 bit

key size → variable (32 to 448) bits

No. of subkeys → 18 (P-array) $P_0, P_1, \ldots P_{17}$

No. of rounds → 16

No. of substitution boxes → 4 s-boxes

having 256 enteries

each is of 32 bit

4) Generation of subkeys

→ 18 subkeys (P[0] .. P[17]) are used for encryption as well as decryption.

→ 18 subkeys are stored in a P-array with each array element being 32-bit entry.

eg: P[0] = "243f6a68" } hexadecimal representation
P[1] = "897gabf3" } of each subkey.

⋮

P[17] = "C97C50dd"

Note

(It) follows feistel structure

fast, compact, simple, secure

↓ executes in less memory

↓ XOR & add operation

↓ variable length key

→ Now, each of the subkey is changed with respect to the i/p key as:

P[0] = P[0] XOR 1st 32 bits of i/p key

P[1] = P[1] XOR 2nd 32 bits of i/p key

⋮

P[13] = P[13] XOR 14th 32 bits of i/p key

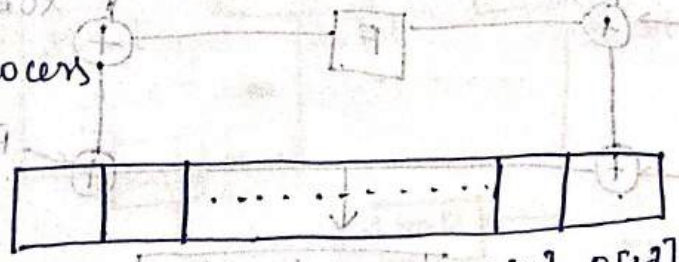
(14×32 = 448 bit key max size of key)

→ if key has less bits then roll over to the 1st 32 bits

P[14] = P[14] XOR 1st 32 bits of key

P[17] = P[17] XOR 4th 32 bits of key

Now, the resultant P-array holds 18 subkeys that is used during the entire encryption process



(18 subkeys)

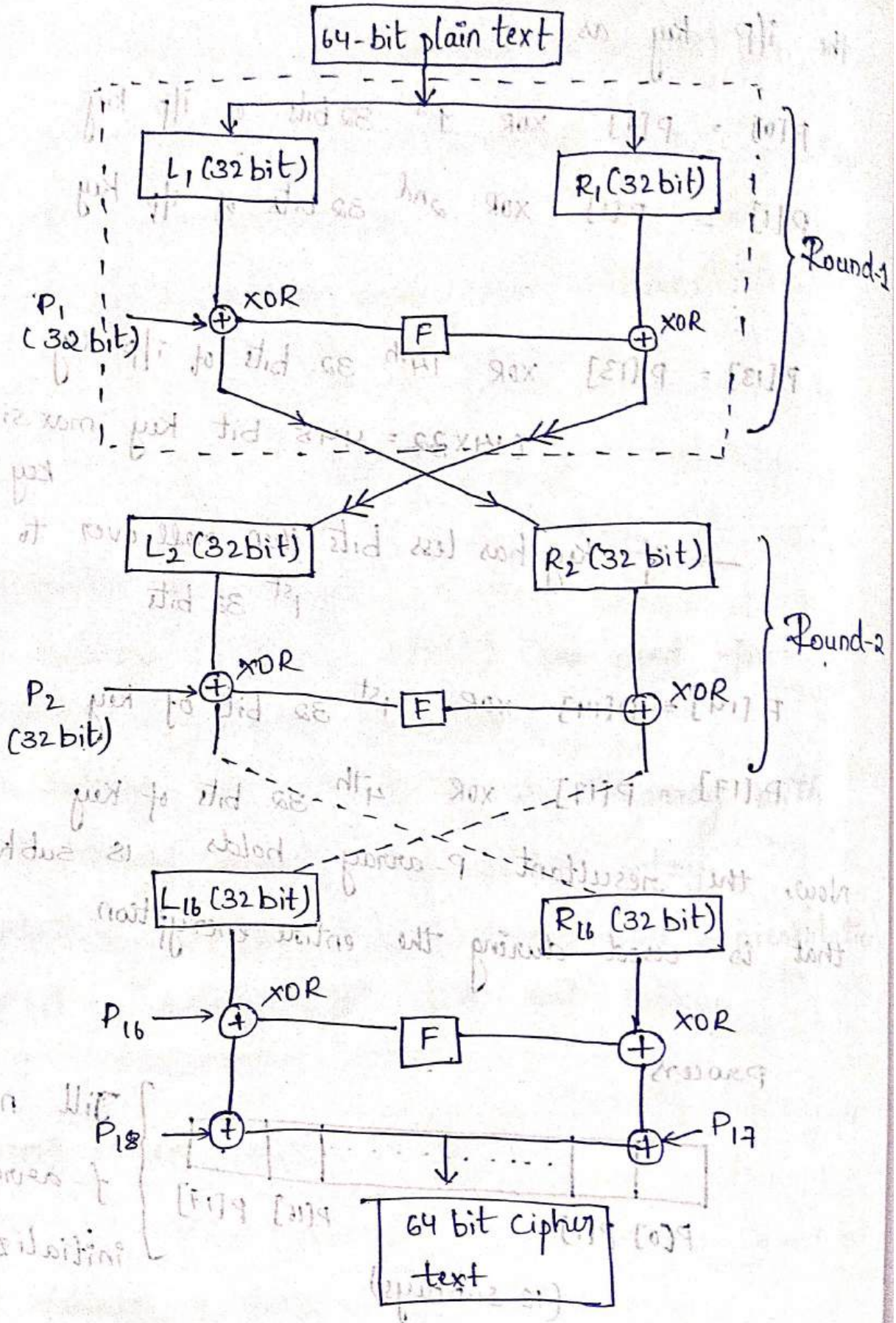
Till now P-array initialized

Step-2 → Initialize substitution boxes

→ 4 S-boxes S[0], S[1], S[2], S[3] used for encryption and decryption

→ have 256 entries (32 bit each)

Step-3 : encryption



Algorithm for encryption of 64-bit Block

① Divide plaintext into 2 blocks 'L' and 'R' of equal sizes (32 bit each)

② for i = 1 to 16

$L = L \oplus P_i$

$R = F(L) \oplus R$

swap L, R

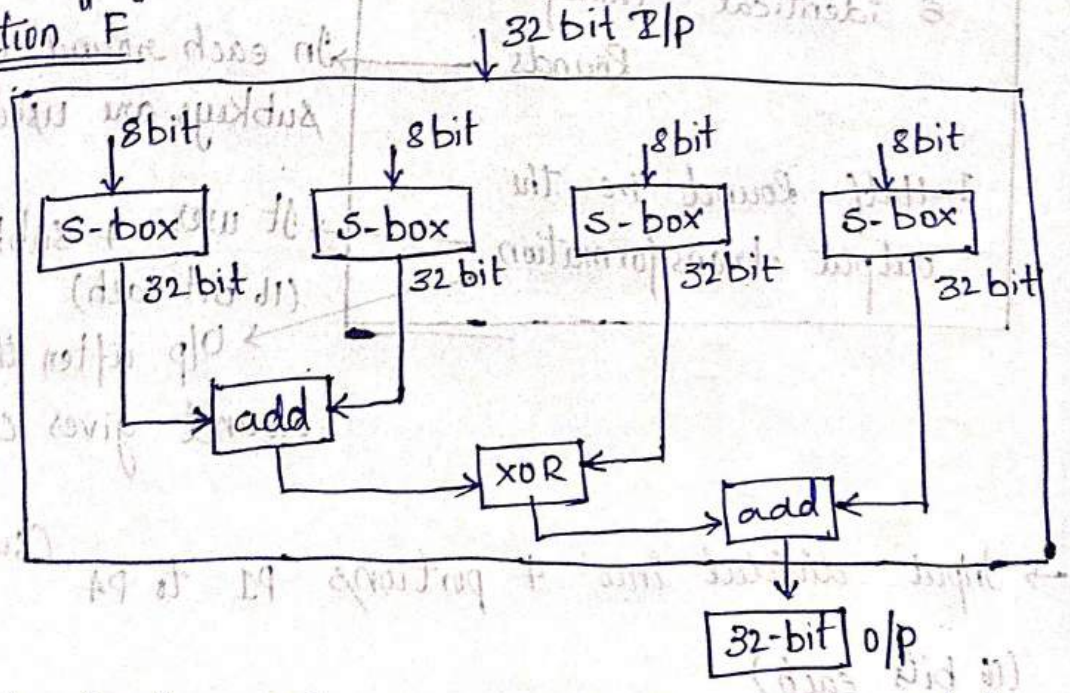
③ undo last swap

④ $R = R \oplus P_{17}$

$L = L \oplus P_{18}$

⑤ Concatenate L and R to get 64-bit cipher text.

Function "F"



↳ Function "f" splits the 32-bit i/p into 4-8bits quarters and 8 bit is given as i/p to each s-box

↳ Each s-box produces 32 bit O/P.

↳ Here, the function add is addition modulo 2^{32}.

IDEA algorithm:

(International Data encryption algorithm)

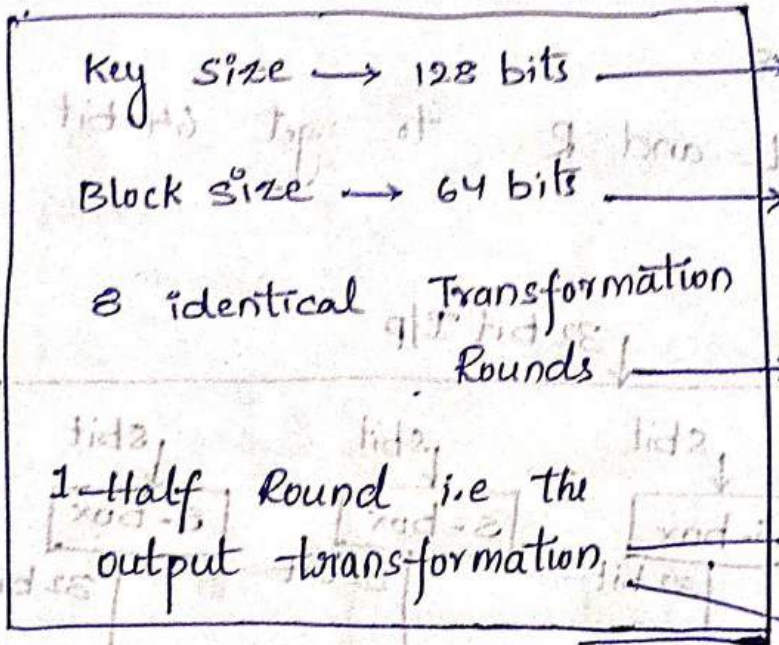
→ Originally called IPES (improved proposed encryption standard)

→ Symmetric key block cipher (designed by James Massey and Xuejia Lai and was first described in 1991)

→ It was intended as a replacement for the DES (data encryption standard)

→ ~~It was intended as a replacement for the DES.~~

→ It is reversible like DES



→ from which we will generate 52 subkeys.

→ In each round, block divided into 4 parts

→ In each round, 6 subkeys are used (16 bit each)

→ It uses 4 subkeys (16 bit each)

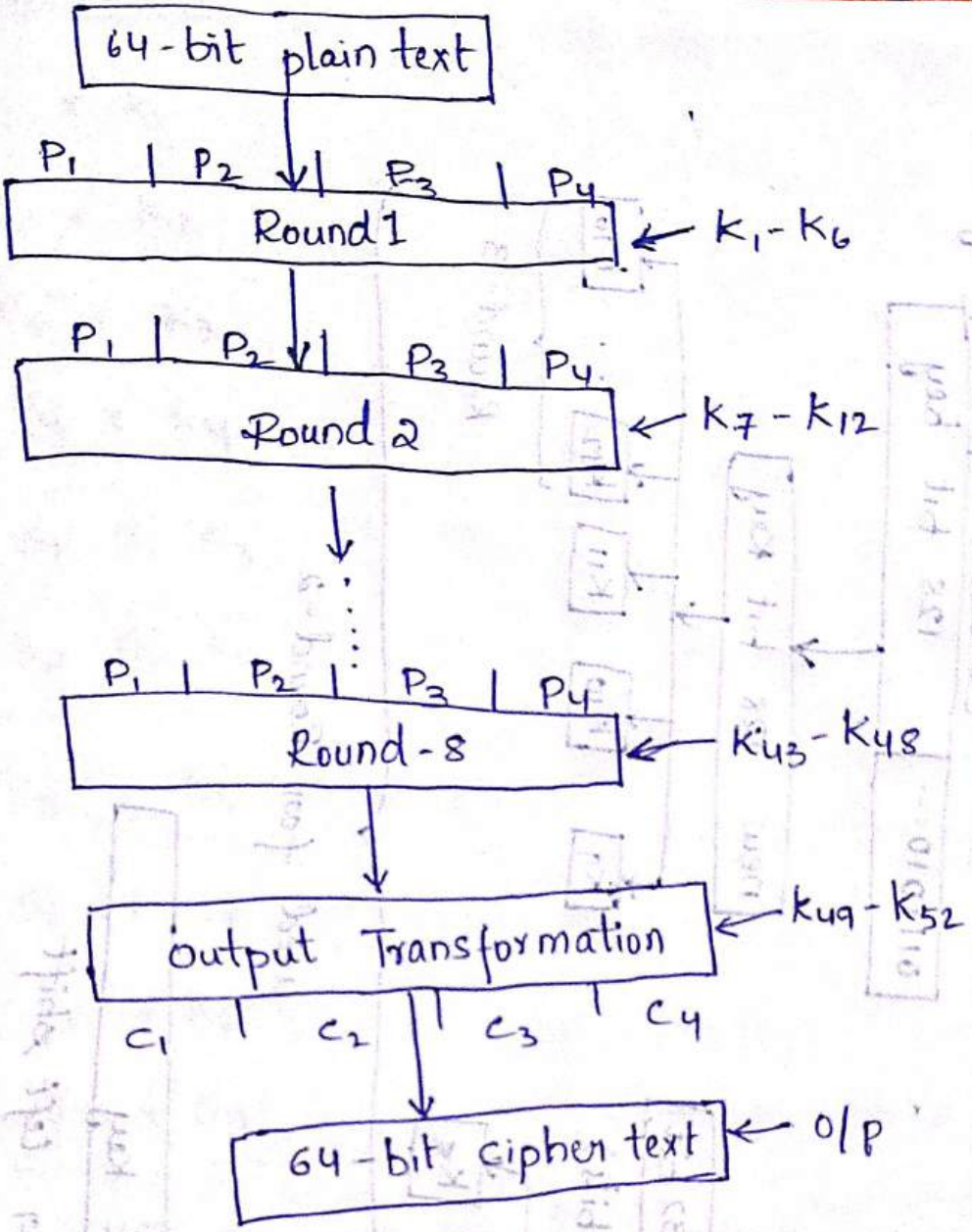
→ O/p after this round gives cipher text. (64 bit)

→ Input divided into 4 portions P1 to P4 (16 bits each)

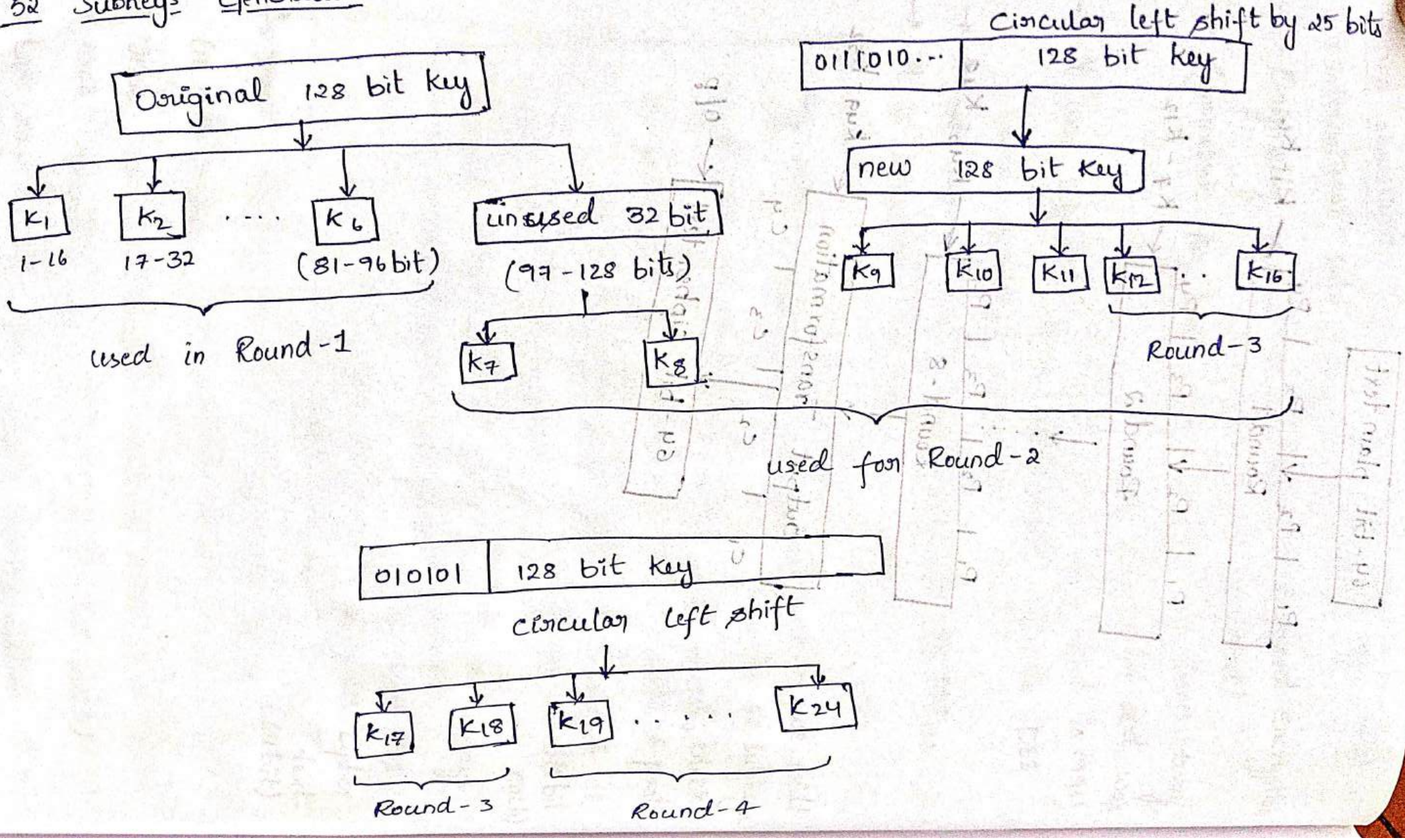
→ There are 8 similar rounds

→ each round uses 6 subkeys (16 bit each)

→ Last round i.e., the output transformation produces the cipher text and uses 4 subkeys (16 bit each)


64-bit plain text
P1 | P2 | P3 | P4
Round 1
K1 - K6
P1 | P2 | P3 | P4
Round 2
K7 - K12
P1 | P2 | P3 | P4
Round - 8
K43 - K48
Output Transformation
K49 - K52
C1 C2 C3 C4
64-bit cipher text
O/P

52 Subkeys Generation:



Single Round Details

$S_1 = P_1 \times K_1$

$S_2 = P_2 + K_2$

$S_3 = P_3 + K_3$

$S_4 = P_4 \times K_4$

$S_5 = S_1 \oplus S_3$

$S_6 = S_2 \oplus S_4$

$S_7 = S_5 \times K_5$

$S_8 = S_6 + S_7$

$S_9 = S_8 \times K_6$

$S_{10} = S_7 + S_9$

$S_{11} = S_1 \oplus S_9 \longrightarrow$ new P_1

$S_{12} = S_3 \oplus S_9 \longrightarrow$ new P_2

$S_{13} = S_2 \oplus S_{10} \longrightarrow$ new P_3 } Swap

$S_{14} = S_4 \oplus S_{10} \longrightarrow$ new P_4

In short

6 subkeys used in 1 round

6 times XOR

4 times multiply modulo

4 times + } Swap value at the end.

Output Trans-formation i.e., one-half round

$R_1 \times K_{49} \longrightarrow C_1$

$R_2 + K_{50} \longrightarrow C_2$

$R_3 + K_{51} \longrightarrow C_3$

$R_4 \times K_{52} \longrightarrow C_4$

→ It takes place at the end of 8th round

→ I/p to this block is a 64-bit value divided into 4-sub blocks (say R_1, R_2, R_3 and R_4)

Block Cipher modes of operation:

→ For different types of messages, we need different modes of operations.

5 modes of operation are:

(i) ECB — electronic codebook mode
(ii) CBC — cipher block chaining mode
(iii) CFB — cipher feedback mode
(iv) OFB — output feedback mode
(v) CTR — counter mode

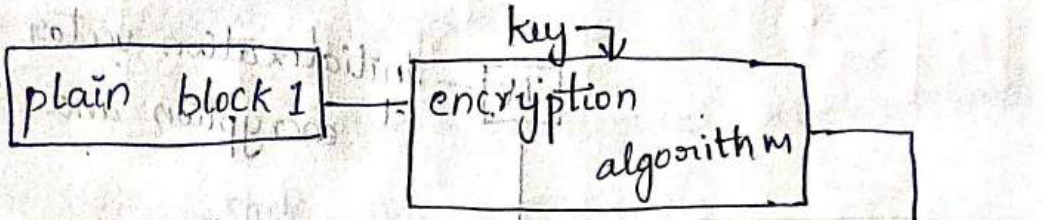
ECB (electronic codebook mode)

* Simplest mode of operation
* plain text is divided into a no. of fixed size block.
* If message is not a multiple of block size, then padding is done.
* Take one block at a time and encrypt it.

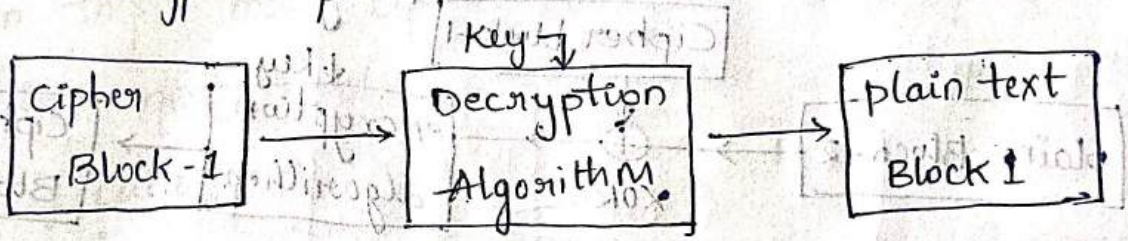
* Same key used for encryption and decryption.

eg: Let block size = 5

plain Text → Hello everyone



→ for decryption process



→ This will happend for all the blocks.

Note

* best for short amount of data, such as a key
* Not secure for lengthy data
* If identical blocks appear, then this mode produces same cipher.

(iii) CBC (Cipher block chaining mode)

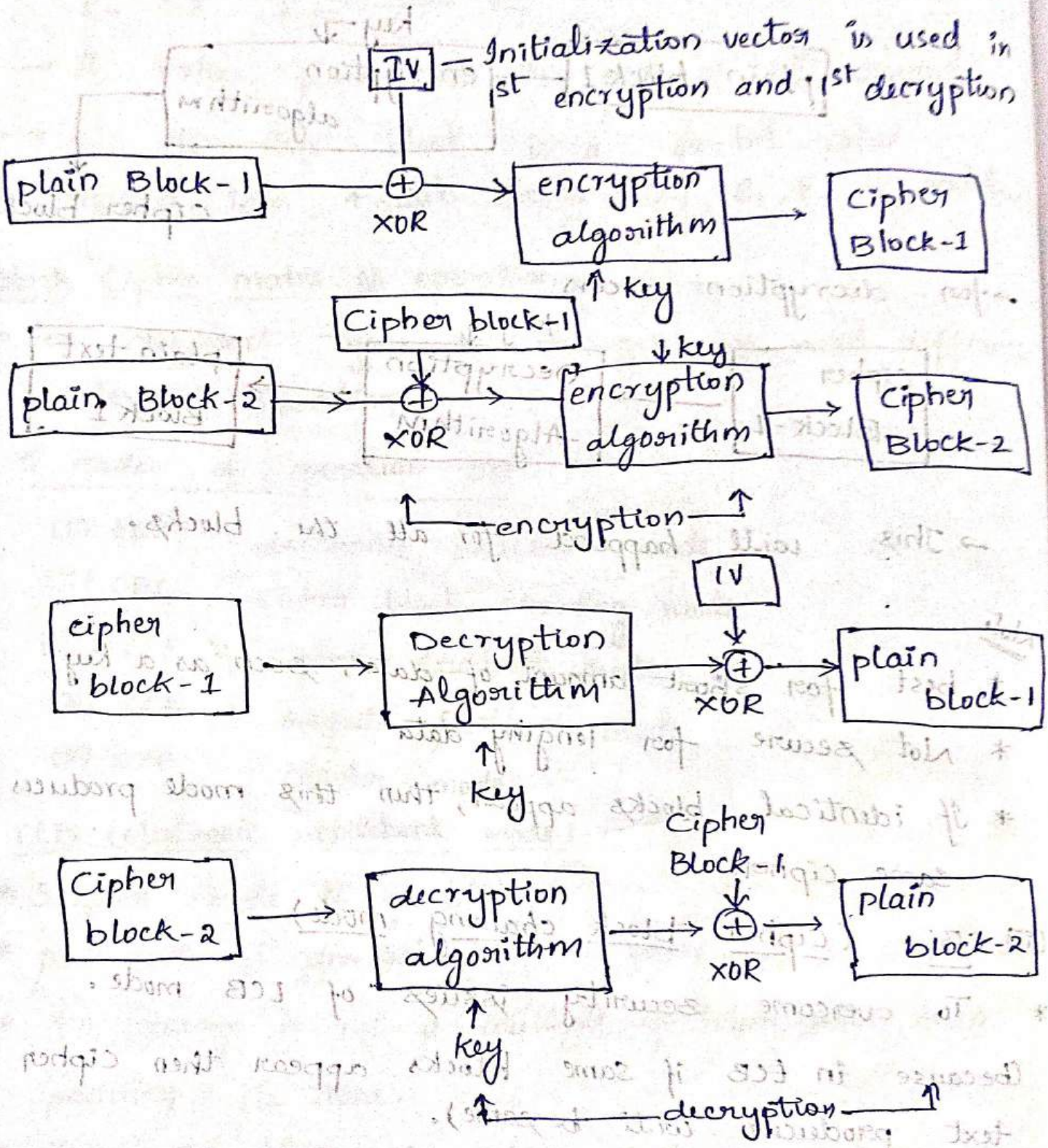
* To overcome security issues of ECB mode.

(because in ECB if same blocks appear then cipher text produced will be same).

* I/p to the encryption algorithm is XOR of the current plain text block and the preceding cipher text block. So, repeating patterns not exposed.

* Same key for encrypt and decrypt

IV is data block of same size



→ IV must be known to both parties, but should be unpredictable by the 3rd parties.

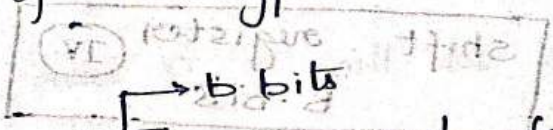
So, we can use ECB encryption to ensure max security

→ Now if we have 2 different blocks it will produce different ciphers

Limitation → If we have 2 identical messags and if we use same IV, cipher will be same.

③ (Cipher Feedback mode) CFB

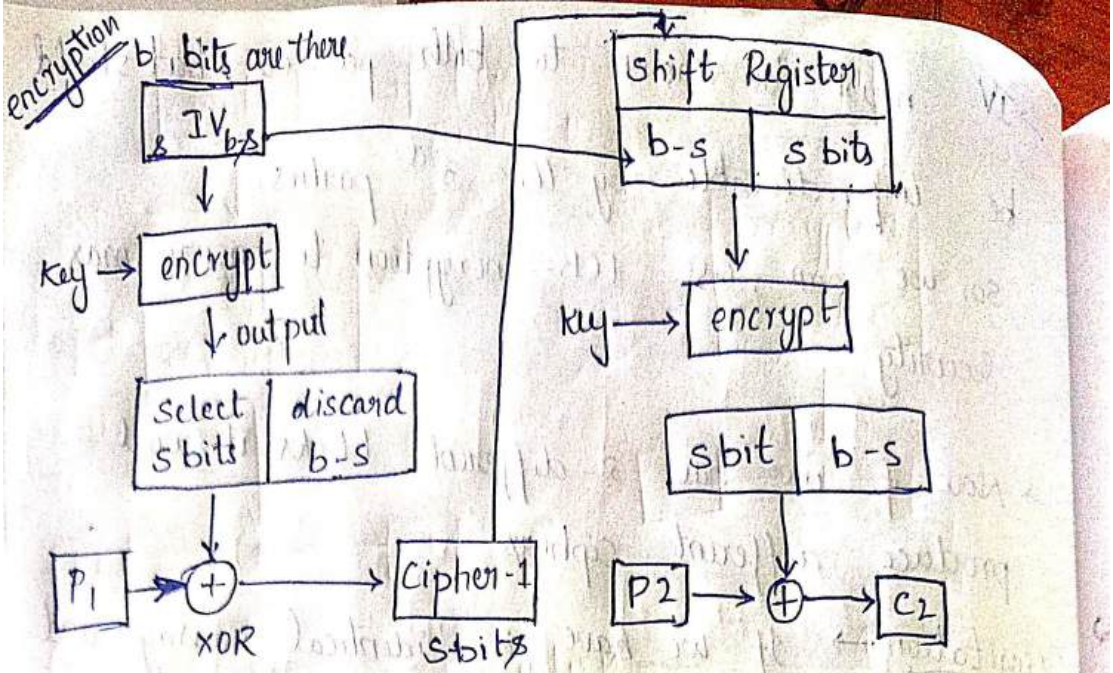
In this mode cipher is given as a feedback to the next block of encryption with some new specifications:



→ 1st an initial vector IV is used for 1st encryption and the o/p bits are divided as set of s and b-s bits

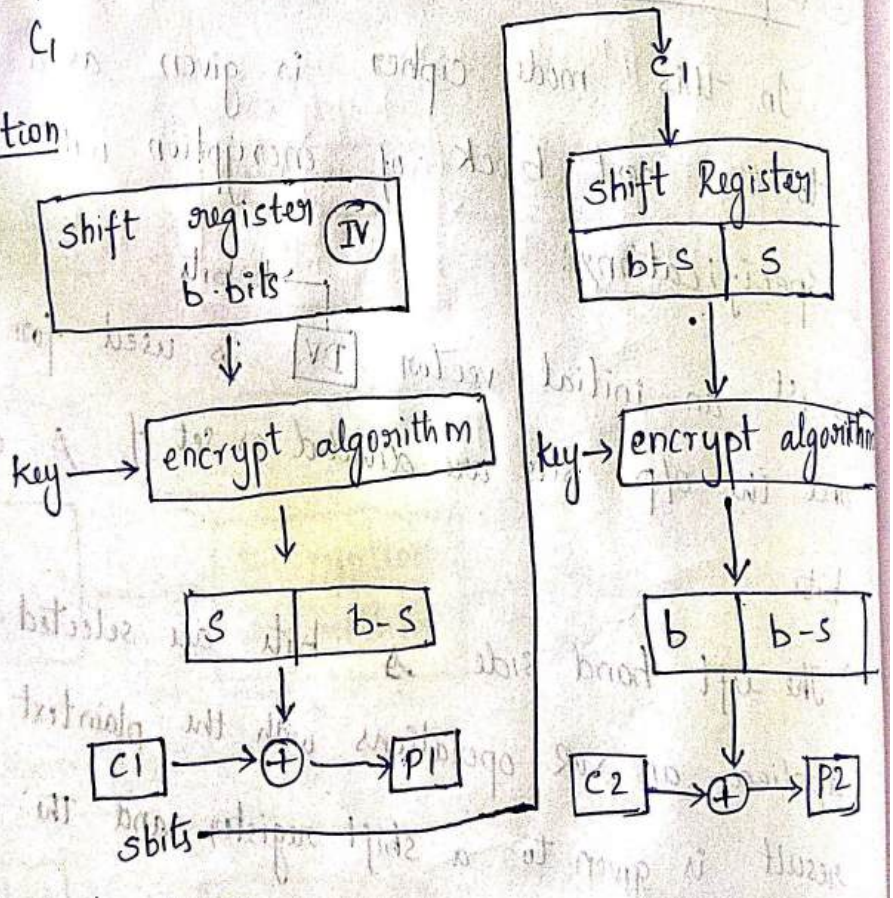
→ The left hand side s bits are selected and are applied an XOR operations with the plaintext bits. The result is given to a shift register and the procuss continues.

→ The plaintext is divided into s bits s can have any value.



→ The (b-s) bits loaded into shift register and a left shift will be done then the s bits loaded from C_1

Decryption

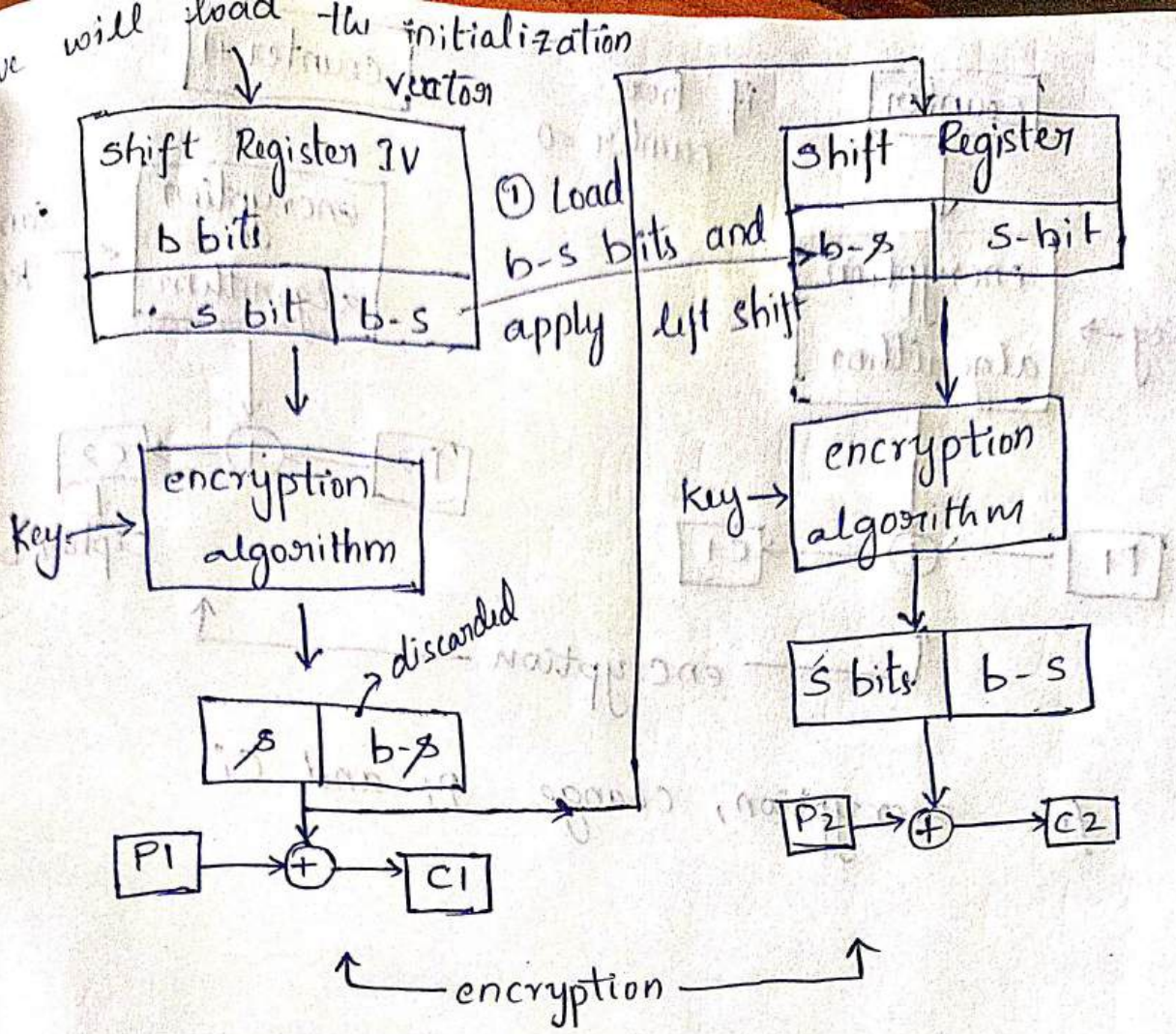


④ OFB (output feedback mode)

→ Similar in structure to CFB

→ The O/p of encryption that is feedback to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register.

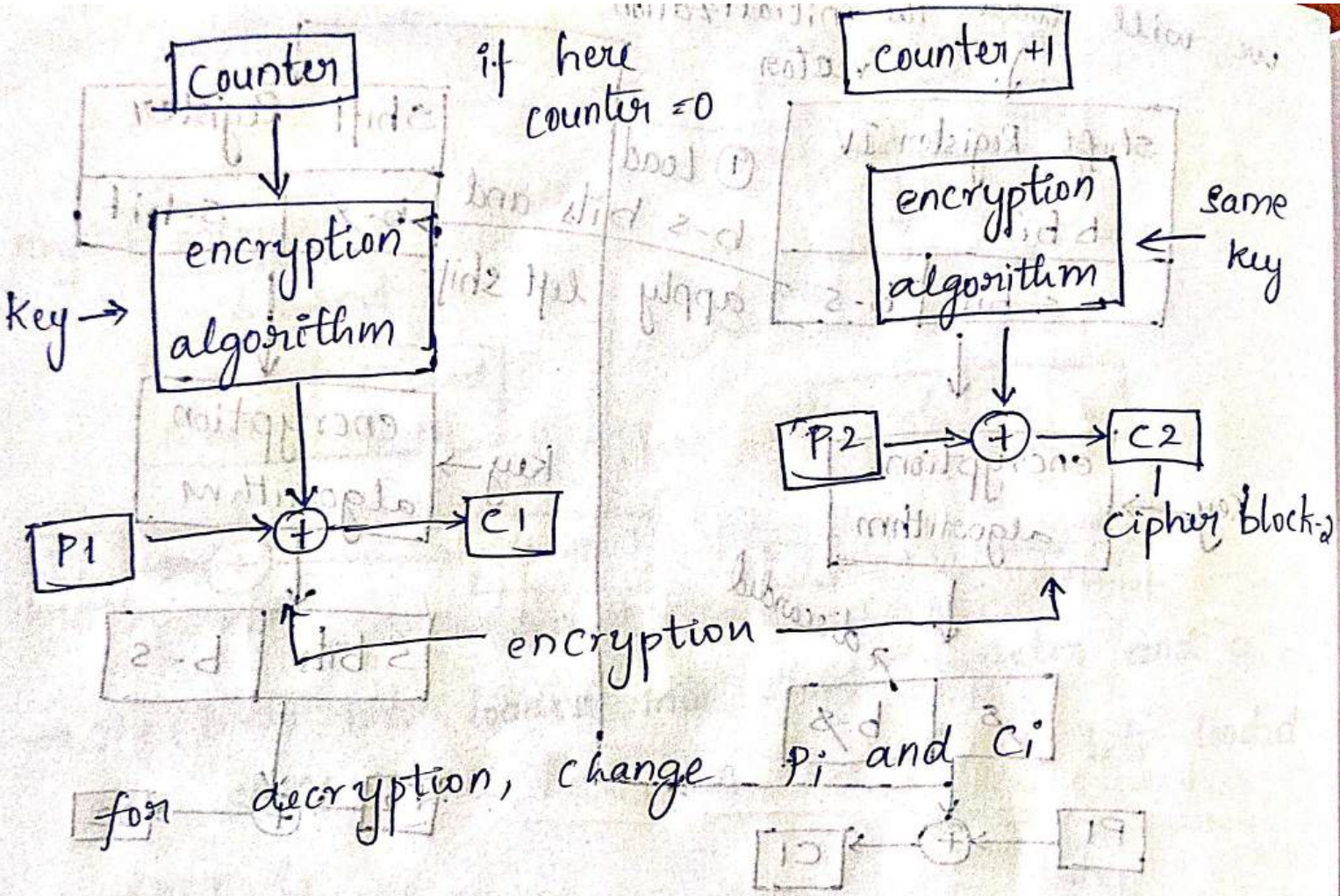
we will load the initialization vector



For decryption, c_j and P_j will be exchanged

⑤ Counter mode CTR:

* Simple and fast
* a counter, equal to the plain text block size is used.
* Counter is initialized to some value and then incremented by 1 for each subsequent block.


Counter
if here counter =0
Counter +1
encryption algorithm
Key
encryption algorithm
same key
P1
C1
P2
C2
cipher block-2
encryption
for decryption, change Pi and Ci