CS8792

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 2 NOTES

UNIT II **SYMMETRIC KEY CRYPTOGRAPHY**

MATHEMATICS OF SYMMETRIC KEY CRYPTOGRAPHY: Algebraic structures - Modular arithmetic-Euclid‟s algorithm- Congruence and matrices - Groups, Rings, Fields- Finite fields- SYMMETRIC KEY CIPHERS: SDES – Block cipher Principles of DES – Strength of DES – Differential and linear cryptanalysis - Block cipher design principles – Block cipher mode of operation – Evaluation criteria for AES – Advanced Encryption Standard - RC4 – Key distribution.

2.1 NUMBER THEORY

- Number Theory is defined as study of integers.

Divisibility:

- If a,b,m are integers.
- If b divides a and if there is no remainder on division then a=mb.
- The notation b / a is commonly used to mean b divides a. (b is a divisor of a)

Properties:

- If a / 1 then a=$\pm$1
- If a / b and b / a then a = $\pm$ b for any b not equal to zero divides 0.
- If a / b and b / c then a / c.
- If b / g and b / h then b / (mg+nh) for integers m and n.
    - If b / g , then g is of the form g=b*g1 for some integer g1.
    - If b / h , then h is of the form h=b*h1 for some integer h1.
    - So mg+nh = mbg1+nbh1=b(mg1+nh1) and therefore b divides mg+nh.

Division Algorithm:

Given any positive integer n and any non negative integer a, if we divide a by n, we get an integer quotient q and an integer remainder e.

$$\mathbf{a=qn+r \quad 0 \leq r < n; \; q=a / n}$$

2.2 ALGEBRAIC STRUCTURES

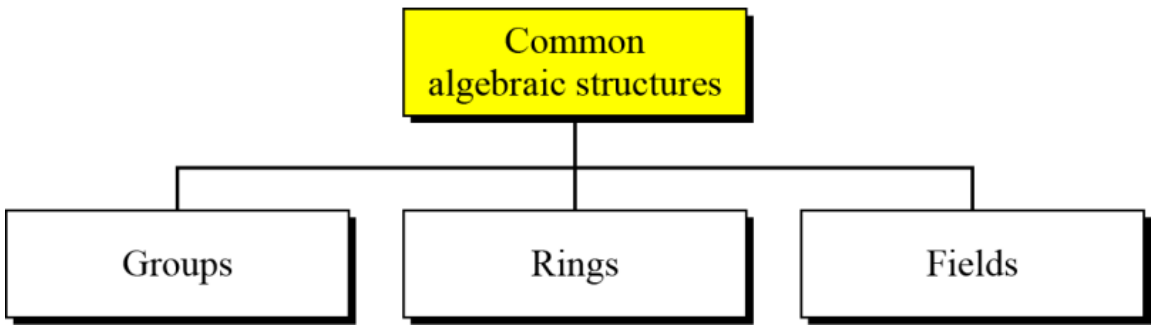
Cryptography requires sets of integers and specific operations that are defined for

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

Operations:

- Addition and Multiplication of numbers.
- Addition and Multiplication of polynomials.
- Addition and Multiplication of matrices.
- Modular arithmetic
- Union and Intersection of sets.
- Composition of Permutations.

Common Algebraic Structure



2.3 MODULAR ARITHMETIC

Modular Arithmetic

The Modulus

If a is an integer and n is a positive integer, we define $a \bmod n$ to be the remainder when a is divided by n. The integer n is called the **modulus**. Thus, for any integer a, we can rewrite Equation (4.1) as follows:

$$a = qn + r \qquad 0 \leq r < n;\ q = \lfloor a/n \rfloor$$

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

Two integers a and b are said to be **congruent modulo** n, if $(a \bmod n) = (b \bmod n)$. This is written as $a \equiv b \pmod{n}$.[2]

$$73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}$$

Note that if $a \equiv 0 \pmod{n}$, then $n|a$.

Properties of Congruences

1. $a \equiv b \pmod{n}$ if $n|(a - b)$.
2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$.
3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ imply $a \equiv c \pmod{n}$.

To demonstrate the first point, if $n|(a - b)$, then $(a - b) = kn$ for some k. So we can write $a = b + kn$. Therefore, $(a \bmod n) =$ (remainder when $b + kn$ is divided by n) = (remainder when b is divided by n) $= (b \bmod n)$.

$23 \equiv 8 \pmod{5}$	because	$23 - 8 = 15 = 5 \times 3$
$-11 \equiv 5 \pmod{8}$	because	$-11 - 5 = -16 = 8 \times (-2)$
$81 \equiv 0 \pmod{27}$	because	$81 - 0 = 81 = 27 \times 3$

Modular Arithmetic Operations

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

We demonstrate the first property. Define $(a \bmod n) = r_a$ and $(b \bmod n) = r_b$. Then we can write $a = r_a + jn$ for some integer j and $b = r_b + kn$ for some integer k. Then

$$\begin{aligned}(a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k + j)n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

Examples

$11 \bmod 8 = 3;\ 15 \bmod 8 = 7$

$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2$

$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$

$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4$

$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$

$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5$

$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$

To find $11^7 \bmod 13$, we can proceed as follows:

$11^2 = 121 \equiv 4 \pmod{13}$

$11^4 = (11^2)^2 \equiv 4^2 \equiv 3 \pmod{13}$

$11^7 = 11 \times 4 \times 3 = 132 = 2 \pmod{13}$

Properties of Modular Arithmetic

set of residues, or residue classes (mod n)

Define the set Z_n as the set of nonnegative integers less than n:

$$Z_n = \{0, 1, \ldots, (n-1)\}$$

precise, each integer in Z_n represents a residue class. We can label the residue classes (mod n) as $[0]$, $[1]$, $[2]$, $\ldots$, $[n-1]$, where

$$[r] = \{a: a \text{ is an integer}, a \equiv r \pmod{n}\}$$

The residue classes (mod 4) are

$[0] = \{\ldots, -16, -12, -8, -4, 0, 4, 8, 12, 16, \ldots\}$

$[1] = \{\ldots, -15, -11, -7, -3, 1, 5, 9, 13, 17, \ldots\}$

$[2] = \{\ldots, -14, -10, -6, -2, 2, 6, 10, 14, 18, \ldots\}$

$[3] = \{\ldots, -13, -9, -5, -1, 3, 7, 11, 15, 19, \ldots\}$

reducing k modulo n.

Finding the smallest nonnegative integer to which k is congruent modulo n

if $(a + b) = (a + c) \pmod{n}$ **then** $b = c \pmod{n}$

$(5 + 23) = (5 + 7) \pmod{8};\ 23 = 7 \pmod{8}$

if $(a \times b) = (a \times c) \pmod{n}$ **then** $b = c \pmod{n}$ **if** a is relatively prime to n

Properties of Modular Arithmetic for Integers in Zn

Commutative Laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x + w) \bmod n$
Associative Laws	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributive Law	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse $(-w)$	For each $w \in Z_n$, there exists a z such that $w + z = 0 \bmod n$

Table 4.2 Arithmetic Modulo 8

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

(b) Multiplication modulo 8

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

(c) Additive and multiplicative inverses modulo 8

2.4 EUCLIDS ALGORITHM

The Euclidean Algorithm

- Simple procedure for determining the greatest common divisor of two positive integers.
- Two integers are relatively prime if their only common positive integer factor is 1

Greatest Common Divisor

- largest integer that divides both a and b
- gcd(0, 0) = 0.

the positive integer c is said to be the greatest common divisor of a and b if

1. c is a divisor of a and of b
2. Any divisor of a and b is a divisor of c

$$\gcd(a, b) = \max[k, \text{such that } k|a \text{ and } k|b]\ \gcd(a, b) = \gcd(|a|, |b|).$$

$$\gcd(60, 24) = \gcd(60, -24) = 12$$

- a and b are relatively prime if gcd(a, b) = 1

Example:

8 and 15 are relatively prime because the positive divisors of 8 are 1, 2, 4, and 8, and the positive divisors of are 1, 3, 5, and 15. So 1 is the only integer on both lists.

Steps

$$\begin{aligned}
a &= q_1 b + r_1 && 0 < r_1 < b \\
b &= q_2 r_1 + r_2 && 0 < r_2 < r_1 \\
r_1 &= q_3 r_2 + r_3 && 0 < r_3 < r_2 \\
&\ \ \vdots && \ \ \vdots \\
r_{n-2} &= q_n r_{n-1} + r_n && 0 < r_n < r_{n-1} \\
r_{n-1} &= q_{n+1} r_n + 0 \\
d &= \gcd(a, b) = r_n
\end{aligned}$$

Example

To find $d = \gcd(a,b) = \gcd(1160718174, 316258250)$		
$a = q_1 b + r_1$	$1160718174 = 3 \times 316258250 + 211943424$	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	$316258250 = 1 \times 211943424 + 104314826$	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	$211943424 = 2 \times 104314826 + 3313772$	$d = \gcd(104314826, 3313772)$

Dividend	Divisor	Quotient	Remainder
$a = 1160718174$	$b = 316258250$	$q_1 = 3$	$r_1 = 211943424$
$b = 316258250$	$r_1 = 211943434$	$q_2 = 1$	$r_2 = 104314826$
$r_1 = 211943424$	$r_2 = 104314826$	$q_3 = 2$	$r_3 = 3313772$

Euclidean Algorithm Revisited

- For any nonnegative integer a and any positive integer b,
- gcd(a, b) = gcd (b, a mod b)
  - Example: gcd(55, 22) = gcd(22, 55 mod 22) = gcd(22, 11) = 11

Recursive function.

```
Euclid(a,b)
      if (b=0) then return a;
      else return Euclid(b, a mod b);
```

The Extended Euclidean Algorithm

calculate the greatest common divisor but also two additional integers and that satisfy the following equation

$$ax + by = d = \gcd(a, b)$$

x and y will have opposite signs

(4.3), and we assume that at each step i we can find integers x_i and y_i that satisfy $r_i = ax_i + by_i$. We end up with the following sequence.

$$\begin{aligned} a &= q_1 b + r_1 & r_1 &= ax_1 + by_1 \\ b &= q_2 r_1 + r_2 & r_2 &= ax_2 + by_2 \\ r_1 &= q_3 r_2 + r_3 & r_3 &= ax_3 + by_3 \\ &\cdot & &\cdot \end{aligned} \qquad \begin{aligned} r_{n-2} &= q_n r_{n-1} + r_n \\ r_{n-1} &= q_{n+1} r_n + 0 \end{aligned} \qquad r_n = ax_n + by_n$$

we can rearrange terms to write

$$r_i = r_{i-2} - r_{i-1} q_i$$

Also, in rows $i - 1$ and $i - 2$, we find the values

$$r_{i-2} = ax_{i-2} + by_{i-2} \quad \text{and} \quad r_{i-1} = ax_{i-1} + by_{i-1}$$

Substituting into Equation (4.8), we have

$$\begin{aligned} r_i &= (ax_{i-2} + by_{i-2}) - (ax_{i-1} + by_{i-1})q_i \\ &= a(x_{i-2} - q_i x_{i-1}) + b(y_{i-2} - q_i y_{i-1}) \end{aligned}$$

But we have already assumed that $r_i = ax_i + by_i$. Therefore,

$$x_i = x_{i-2} - q_i x_{i-1} \quad \text{and} \quad y_i = y_{i-2} - q_i y_{i-1}$$

Let us now look at an example with relatively large numbers to see the power of this algorithm:

To find $d = \gcd(a,b) = \gcd(1160718174, 316258250)$				
$a = q_1 b + r_1$	1160718174	= 3 × 316258250 +	211943424	$d = \gcd(316258250, 211943424)$
$b = q_2 r_1 + r_2$	316258250	= 1 × 211943424 +	104314826	$d = \gcd(211943424, 104314826)$
$r_1 = q_3 r_2 + r_3$	211943424	= 2 × 104314826 +	3313772	$d = \gcd(104314826, 3313772)$
$r_2 = q_4 r_3 + r_4$	104314826	= 31 × 3313772 +	1587894	$d = \gcd(3313772, 1587894)$
$r_3 = q_5 r_4 + r_5$	3313772	= 2 × 1587894 +	137984	$d = \gcd(1587894, 137984)$
$r_4 = q_6 r_5 + r_6$	1587894	= 11 × 137984 +	70070	$d = \gcd(137984, 70070)$
$r_5 = q_7 r_6 + r_7$	137984	= 1 × 70070 +	67914	$d = \gcd(70070, 67914)$
$r_6 = q_8 r_7 + r_8$	70070	= 1 × 67914 +	2156	$d = \gcd(67914, 2156)$
$r_7 = q_9 r_8 + r_9$	67914	= 31 × 2516 +	1078	$d = \gcd(2156, 1078)$
$r_8 = q_{10} r_9 + r_{10}$	2156	= 2 × 1078 +	0	$d = \gcd(1078, 0) = 1078$
Therefore, $d = \gcd(1160718174, 316258250) = 1078$				

The Extended Euclidean Algorithm

Given two integers a and b, we often need to find other two integers, s and t, such that

$$s \times a + t \times b = \gcd(a, b)$$

The **extended Euclidean algorithm** can calculate the gcd (a, b) and at the same time

1	28	21	7	0	1	−1	1	−5	6
3	21	7	0	1	−1	4	−5	6	−23
	7	0		−1	4		6	−23	

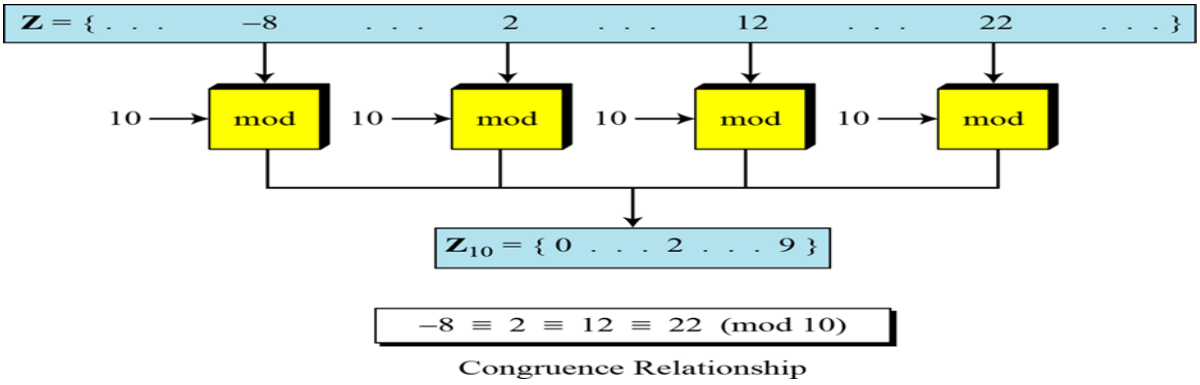
We get gcd (161, 28) = 7, $s = -1$ and $t = 6$. The answers can be tested because we have

$$(-1) \times (161) + 6 \times (28) = (7)$$

22 mod 10 = 2

In modular arithmetic 2, 12 and 22 are called congruent mod10

Concepts of C



2.5 CONGRUENCE AND MATRICES

In cryptography we need to handle matrices

Definition

A matrix of size l x m

$$\text{Matrix } \mathbf{A}: \quad \overset{l \text{ rows}}{\left[\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{array}\right]} \quad \overset{m \text{ columns}}{}$$

Examples of matrices

$$\begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix}$$
Row matrix

$$\begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}$$
Column matrix

$$\begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix}$$
Square matrix

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$
0

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
I

Operations and Relations

An example of addition and subtraction.

Addition and subtraction of matrices

$$\begin{bmatrix} 12 & 4 & 4 \\ 11 & 12 & 30 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} + \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

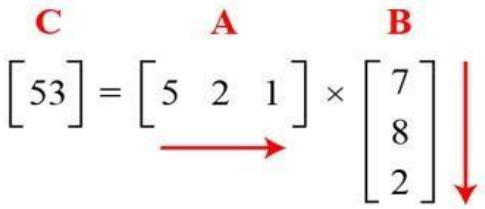
C = A + B

$$\begin{bmatrix} -2 & 0 & -2 \\ -5 & -8 & 10 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} - \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

D = A − B

Figure shows the product of a row matrix (1 × 3) by a column matrix (3 × 1). The result is a matrix of size 1 × 1.

Figure *Multiplication of a row matrix by a column matrix*



In which: $53 = 5 \times 7 + 2 \times 8 + 1 \times 2$

DETERMINANT

The determinant of a square matrix A of size m × m denoted as det (A) is a scalar calculated recursively as shown below:

1. If $m = 1$, $\det(\mathbf{A}) = a_{11}$
2. If $m > 1$, $\det(\mathbf{A}) = \sum_{\cdot\,\cdot\,\cdot} (-1)^{i+j} \times a_{ij} \times \det(\mathbf{A}_{ij})$

Where $\mathbf{A}_{ij}$ is a matrix obtained from $\mathbf{A}$ by deleting the ith row and jth column.

Note

The determinant is defined only for a square matrix.

Figure below shows how we can calculate the determinant of a 2 × 2 matrix based on the determinant of a 1 × 1 matrix.

Figure *Calculating the determinant of a 2 ´ 2 matrix*

$$\det\begin{bmatrix} 5 & 2 \\ 3 & 4 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det[\,4\,] + (-1)^{1+2} \times 2 \times \det[\,3\,] \longrightarrow 5 \times 4 - 2 \times 3 = 14$$

or $$\det\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}$$

Figure below shows the calculation of the determinant of a 3 × 3 matrix.

Figure *Calculating the determinant of a 3 ´ 3 matrix*

$$\det\begin{bmatrix} 5 & 2 & 1 \\ 3 & 0 & -4 \\ 2 & 1 & 6 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det\begin{bmatrix} 0 & -4 \\ 1 & 6 \end{bmatrix} + (-1)^{1+2} \times 2 \times \det\begin{bmatrix} 3 & -4 \\ 2 & 6 \end{bmatrix} + (-1)^{1+3} \times 1 \times \det\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

$$= (+1) \times 5 \times (+4) \quad + \quad (-1) \times 2 \times (24) \quad + \quad (+1) \times 1 \times (3) = -25$$

2.6 GROUPS, RINGS,FIELDS,FINITE FIELDS

Groups, rings, and fields are the fundamental elements of a branch of mathematics known as abstract algebra, or modern algebra. In abstract algebra, we are concerned with sets on whose elements we can operate algebraically; that is, we can combine two elements of the set, perhaps in several ways, to obtain a third element of the set. These operations are subject to specific rules, which define the nature of the set. By convention, the notation for the two principal classes of operations on set elements is usually the same as the notation for addition and multiplication on ordinary numbers. However, it is important to note that, in abstract algebra, we are not limited to ordinary arithmetical operations. All this should become clear as we proceed.

Groups

A **group** G, sometimes denoted by $\{G, \cdot\}$, is a set of elements with a binary operation denoted by $\cdot$ that associates to each ordered pair (a, b) of elements in G an element $(a \cdot b)$ in G, such that the following axioms are obeyed:[4]

(A1) Closure:	If a and b belong to G, then $a \cdot b$ is also in G.
(A2) Associative:	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all a, b, c in G.
(A3) Identity element:	There is an element e in G such that $a \cdot e = e \cdot a = a$ for all a in G.
(A4) Inverse element:	For each a in G, there is an element a' in G such that $a \cdot a' = a' \cdot a = e$.

If a group has a finite number of elements, it is referred to as a **finite group**, and the **order** of the group is equal to the number of elements in the group. Otherwise, the group is an **infinite group**.

A group is said to be **abelian** if it satisfies the following additional condition:

(A5) Commutative: $a \cdot b = b \cdot a$ for all a, b in G.

The set of integers (positive, negative, and 0) under addition is an abelian group. The set of nonzero real numbers under multiplication is an abelian group. The set S_n from the preceding example is a group but not an abelian group for $n > 2$.

When the group operation is addition, the identity element is 0; the inverse element of a is $-a$; and subtraction is defined with the following rule: $a - b = a + (-b)$.

CYCLIC GROUP We define exponentiation within a group as a repeated application of the group operator, so that $a^3 = a \cdot a \cdot a$. Furthermore, we define $a^0 = e$ as the identity element, and $a^{-n} = (a')^n$, where a' is the inverse element of a within the group. A group G is **cyclic** if every element of G is a power a^k (k is an integer) of a fixed element $a \in G$. The element a is said to **generate** the group G or to be a **generator** of G. A cyclic group is always abelian and may be finite or infinite.

The additive group of integers is an infinite cyclic group generated by the element 1. In this case, powers are interpreted additively, so that n is the nth power of 1.

Rings

A **ring** R, sometimes denoted by $\{R, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*,[6] such that for all a, b, c in R the following axioms are obeyed.

(A1–A5) R is an abelian group with respect to addition; that is, R satisfies axioms A1 through A5. For the case of an additive group, we denote the identity element as 0 and the inverse of a as $-a$.

(M1) Closure under multiplication: If a and b belong to R, then ab is also in R.

(M2) Associativity of multiplication: $a(bc) = (ab)c$ for all a, b, c in R.

(M3) Distributive laws: $a(b + c) = ab + ac$ for all a, b, c in R.

$(a + b)c = ac + bc$ for all a, b, c in R.

In essence, a ring is a set in which we can do addition, subtraction $[a - b = a + (-b)]$, and multiplication without leaving the set.

With respect to addition and multiplication, the set of all n-square matrices over the real numbers is a ring.

A ring is said to be **commutative** if it satisfies the following additional condition:

(M4) Commutativity of multiplication: $ab = ba$ for all a, b in R.

Let S be the set of even integers (positive, negative, and 0) under the usual operations of addition and multiplication. S is a commutative ring. The set of all n-square matrices defined in the preceding example is not a commutative ring.

The set Z_n of integers $\{0, 1, \ldots, n-1\}$, together with the arithmetic operations modulo n, is a commutative ring (Table 4.3).

Next, we define an **integral domain**, which is a commutative ring that obeys the following axioms.

(M5) Multiplicative identity: There is an element 1 in R such that $a1 = 1a = a$ for all a in R.

(M6) No zero divisors: If a, b in R and $ab = 0$, then either $a = 0$ or $b = 0$.

Let S be the set of integers, positive, negative, and 0, under the usual operations of addition and multiplication. S is an integral domain.

Fields

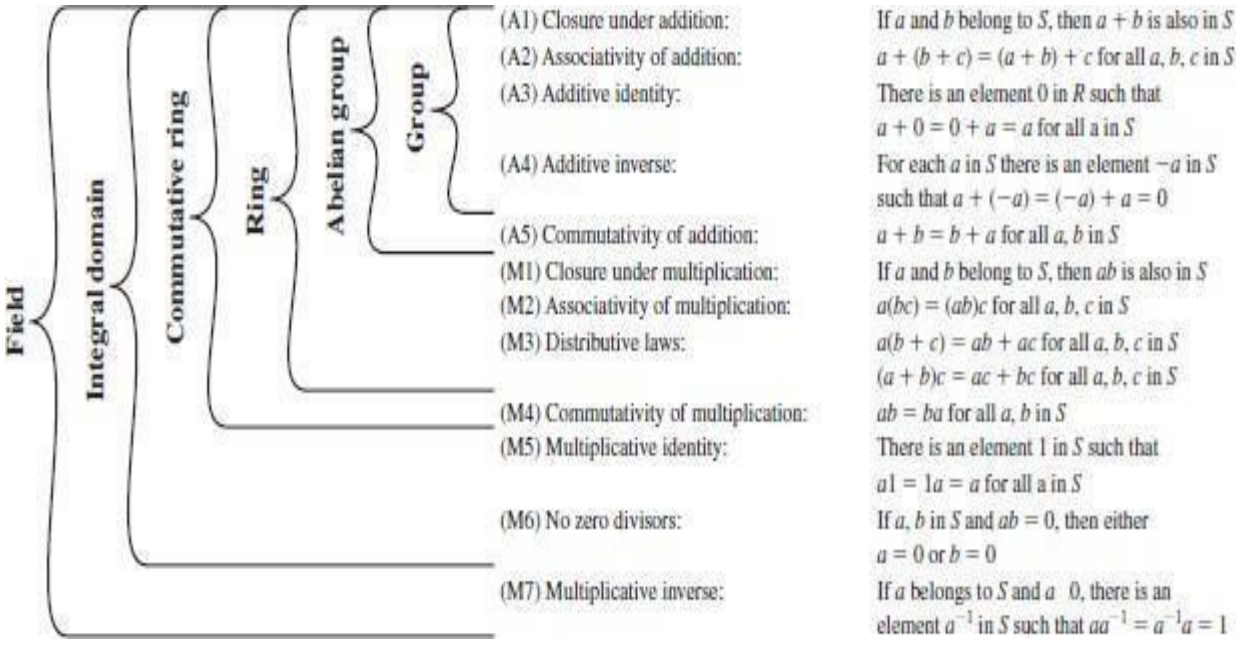
A **field** F, sometimes denoted by $\{F, +, \times\}$, is a set of elements with two binary operations, called *addition* and *multiplication*, such that for all a, b, c in F the following axioms are obeyed.

(A1–M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(M7) Multiplicative inverse: For each a in F, except 0, there is an element a^{-1} in F such that $aa^{-1} = (a^{-1})a = 1$.

In essence, a field is a set in which we can do addition, subtraction, multiplication, and division without leaving the set. Division is defined with the following rule: $a/b = a(b^{-1})$.

Familiar examples of fields are the rational numbers, the real numbers, and the complex numbers. Note that the set of all integers is not a field, because not every element of the set has a multiplicative inverse; in fact, only the elements 1 and –1 have multiplicative inverses in the integers.



Summarize the Axioms that defines Groups, Rings and Fields

FINITE FIELDS OF THE FORM GF(p)

The finite field of order p^n is generally written $GF(p^n)$; GF stands for Galois field, in honor of the mathematician who first studied finite fields. Two special cases are of interest for our purposes. For $n = 1$, we have the finite field $GF(p)$; this finite field has a different structure than that for finite fields with $n > 1$ and is studied in this section. In Section 4.7, we look at finite fields of the form $GF(2^n)$.

Finite Fields of Order p

For a given prime, p, we define the finite field of order p, $GF(p)$, as the set Z_p of integers $\{0, 1, \ldots, p - 1\}$ together with the arithmetic operations modulo p.

Recall that we showed in Section 4.3 that the set Z_n of integers $\{0, 1, \ldots, n - 1\}$, together with the arithmetic operations modulo n, is a commutative ring (Table 4.3). We further observed that any integer in Z_n has a multiplicative inverse if and only if that integer is relatively prime to n [see discussion of Equation (4.5)].[7] If n is prime, then all of the nonzero integers in Z_n are relatively prime to n, and therefore there exists a multiplicative inverse for all of the nonzero integers in Z_n. Thus, for Z_p we can add the following properties to those listed in Table 4.3:

Multiplicative inverse (w^{-1})	For each $w \in Z_p$, $w \neq 0$, there exists a $z \in Z_p$ such that $w \times z \equiv 1 \pmod p$

The simplest finite field is GF(2). Its arithmetic operations are easily summarized:

+	0	1
0	0	1
1	1	0

Addition

×	0	1
0	0	0
1	0	1

Multiplication

w	$-w$	w^{-1}
0	0	—
1	1	1

Inverses

In this case, addition is equivalent to the exclusive-OR (XOR) operation, and multiplication is equivalent to the logical AND operation.

2.7 SYMMETRIC KEY CIPHERS

The Feistel Cipher Structure

- Feistel proposed a structure which alternates substitution and permutation. This follows the concept given by claude Shannon to produce cipher that alternates confusion and diffusion.
- **Diffusion**: making each plaintext bit affect as many cipher text bits as possible.
- **Confusion**: making the relationship between the encryption key and the cipher text as complex as possible.

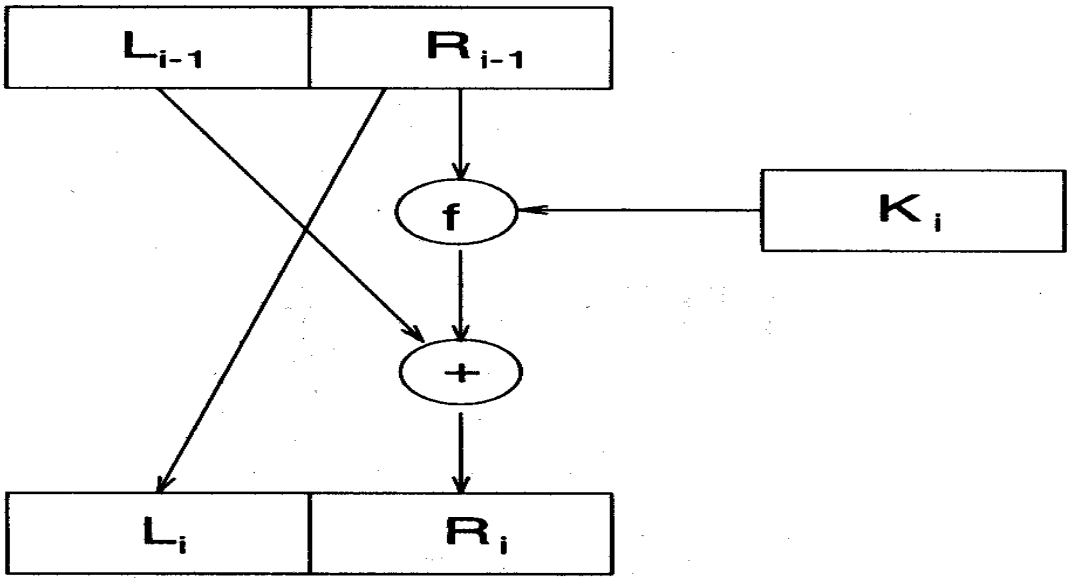
2.7.1.1 Input: a data block and a key

2.7.1.2 Partition the data block into two halves L and R.

2.7.1.3 **Go through a number of rounds**.

2.7.1.4 In each round, R does not change and L goes through an operation that depends on R and a round key derived from the key.

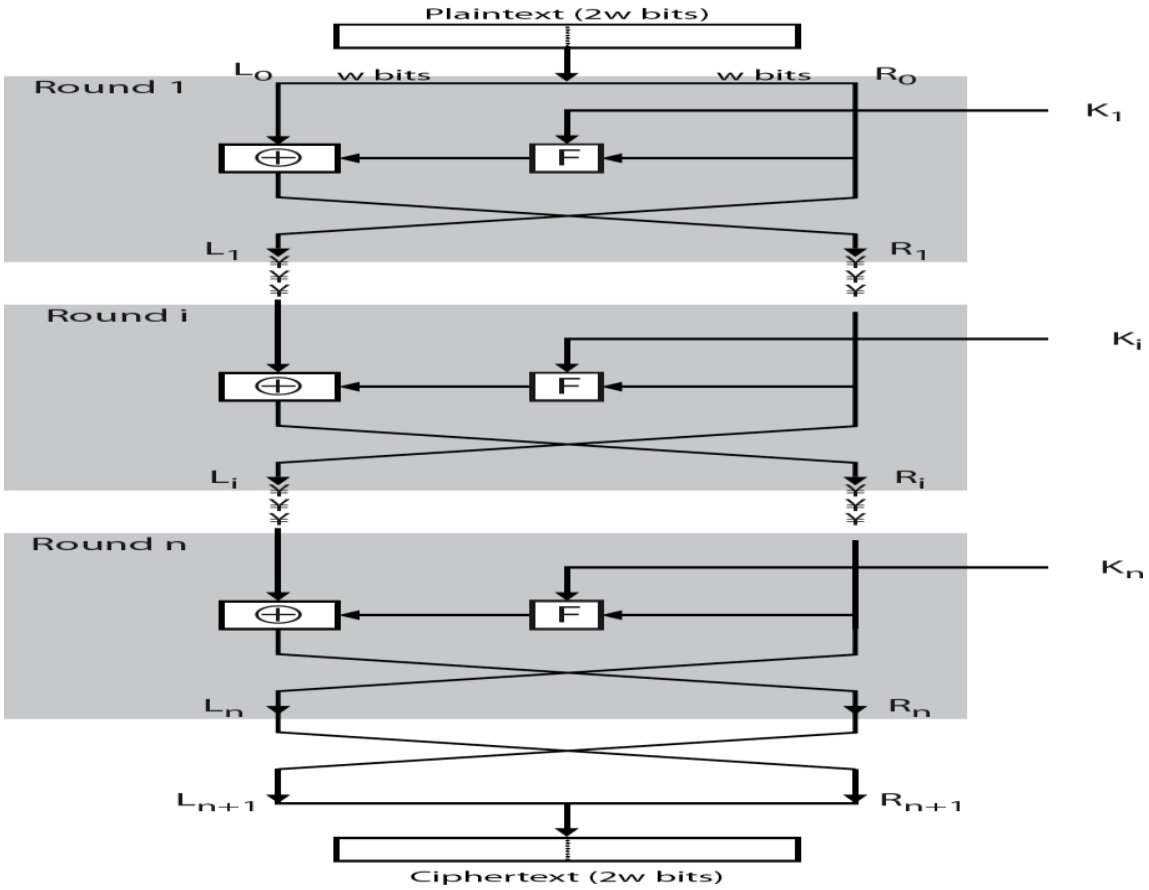
Feistel Cipher (Single round)



Li = Ri-1

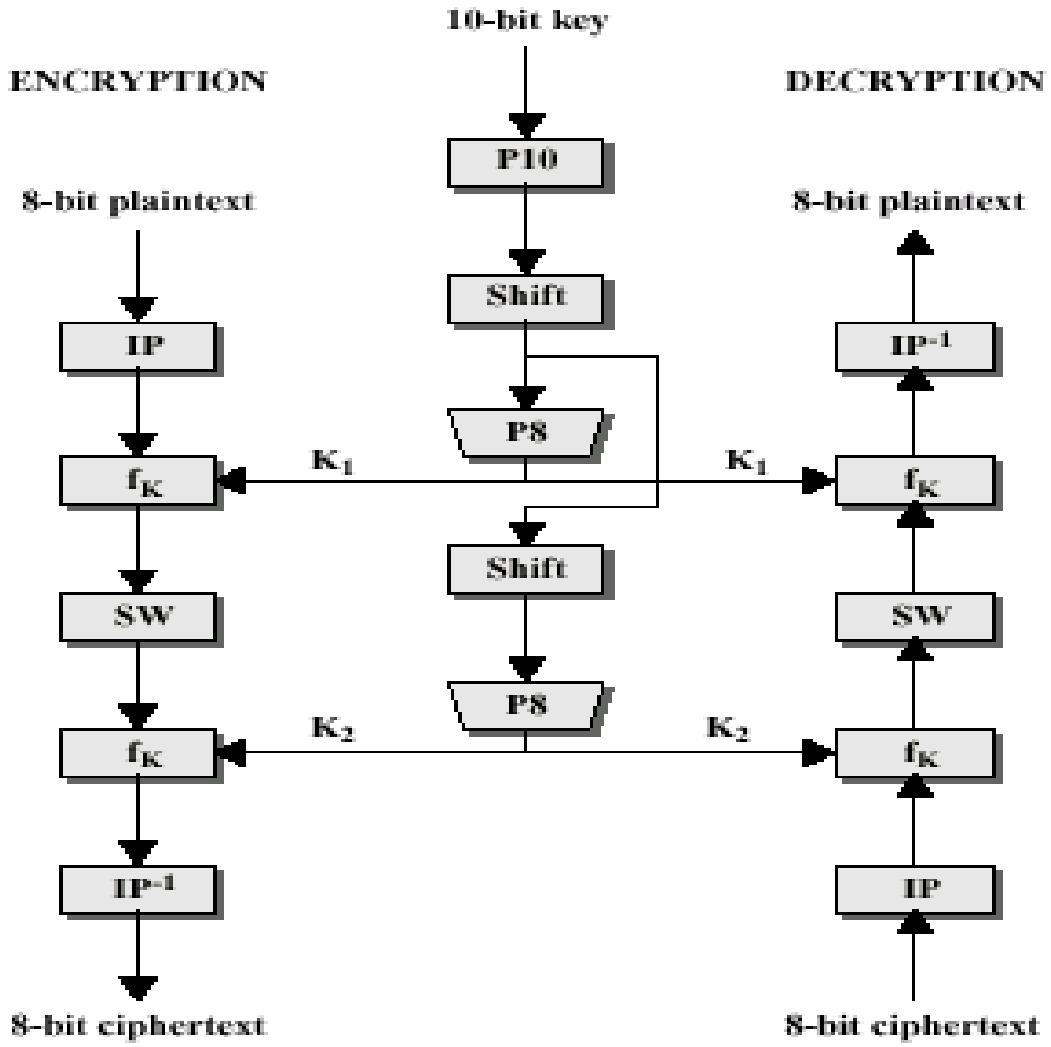
Ri = Li-1 x F(Ri-1, Ki)

The Feistel Cipher Structure



2.8 SIMPLE DES

- The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977.
- **Encryption:** Takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8-bit of cipher.
- **Decryption:**Takes an 8-bit block of cipher and the same 10-bit key as input and produces an 8-bit of original plaintext.
- Both substitution and transposition operations are used
- It is a complex, multi-phase algorithm
- **Five Functions to Encrypt**
    - IP: Initial permutation
    - fk: Key-dependent scrambler ((complex) function))
        - Use a 8-bit key
        - Perform both permutation and substitution
    - SW (simple permutation function)
        - Swap the two halves of data
    - fk again (different key)
    - IP-1: Inverse permutation



- A permutation function that is the inverse of the initial permutation
- The function fk takes as input not only the data passing through the encryption algorithm, but also an 8-bit key. Here a 10-bit key is used from which two 8-bit subkeys are generated.
- The key is first subjected to a permutation (P10). Then a shift operation is performed. The output of the shift operation then passes through a permutation function that produces an 8-bit output (P8) for the first subkey (K1).
- The output of the shift operation also feeds into another shift and another instance of P8 to produce the second subkey (K2).

- The encryption algorithm can be expressed as a composition composition1 of functions:

  IP-1 o f K2 o SW o fk1 o IP, which can also be written as
  Ciphertext = IP-1 (fK2 (SW (fk1 (IP (plaintext))))) Where
  K1 = P8 (Shift (P10 (Key)))
  K2 = P8 (Shift (shift (P10 (Key))))
  Decryption can be shown as Plaintext = IP-1 (fK1 (SW (fk2 (IP (ciphertext)))))

S-DES Key Generation

- S-DES depends on the use of a 10-bit key shared between sender and receiver. From this key, two 8-bit sub keys are produced for use in particular stages of the encryption and decryption algorithm below Figure

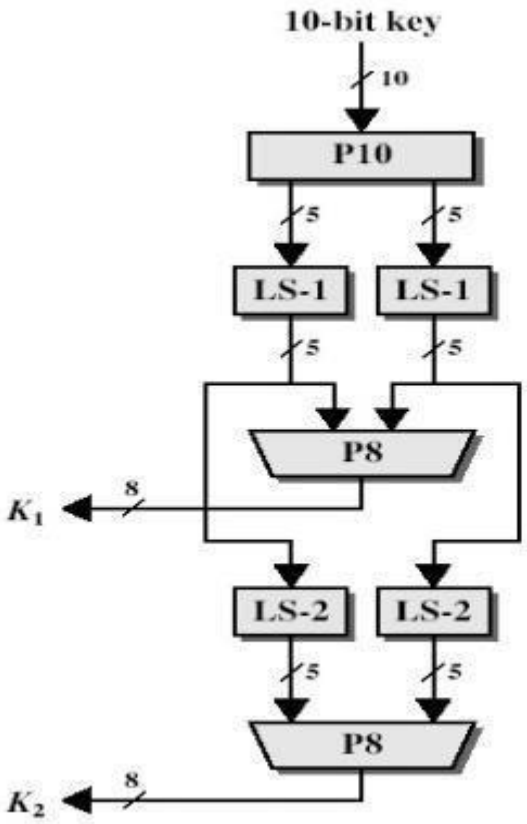


Figure S-DES Key Generation

- First, permute the key in the following fashion. Let the 10-bit key be designated as (k1, K2, k3, k4, k5, k6, k7, k8, k9, k10).
- Then the permutation P10 is defined as:
  P10 (k1, K2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, K2, k7, k4, k10 10, k1, k9,k8, k6).

P10									
3	5	2	7	4	10	1	9	8	6

P10 can be concisely defined by the display

This table is read from left to right; each position in the table gives the identity of the input bit that produces the output bit in that position. So, the first output bit is bit 3 of the input; the second output bit is bit 5 of the input, and so on.

Example

The 10 bit key is (1010000010), now find the permutation from P10 for this key so it becomes (10000 01100).

Next, perform a circular left shift (LS-1), or rotation, separately on the first five bits and the second five bits. In our example, the result is (00001 11000).

Next, apply P8, which picks out and permutes 8 of the 10 bits according to the following rule:

P8							
6	3	7	4	8	5	10	9

So, The result is subkey 1 (K1). In our example, this yield (10100100).

Then go back to the pair of 5-bit strings produced by the two LS-1 functions and performs a circular left shift of 2 bit positions on each string. In our example, the value (00001 11000) becomes (00100 00011).

Finally, P8 is applied again to produce K2. In our example, the result is (01000011).

S-DES Encryption

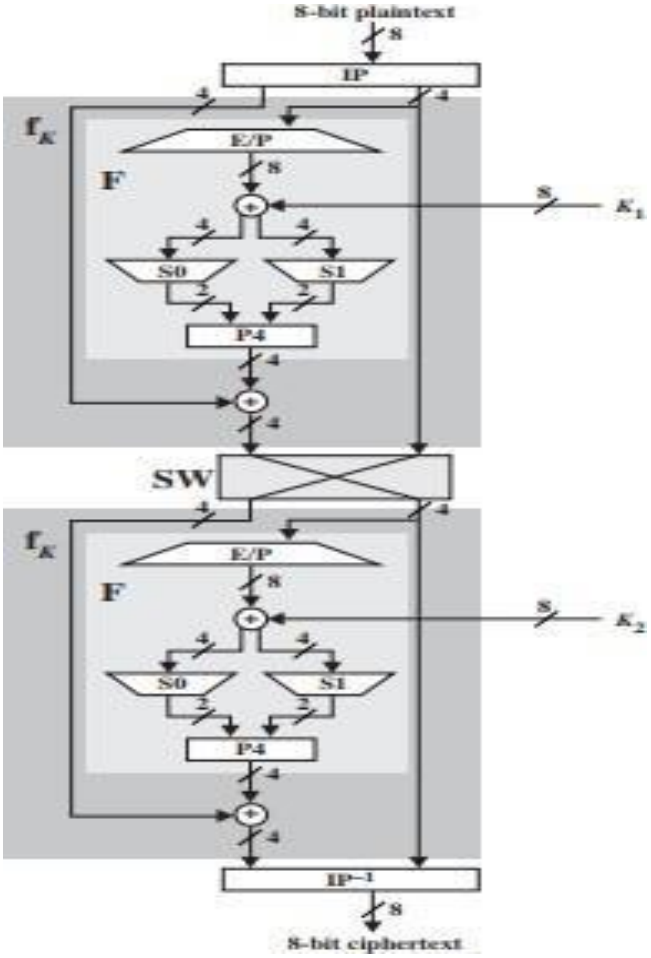
Encryption involves the sequential application of five functions.

1. Initial Permutations

The input to the algorithm is an 8-bit block of plaintext, which we first permute using the IP function

IP							
2	6	3	1	4	8	5	7

The plaintext is
10111101
Permutated
output is 01111110



2. The Function fk

The most complex component of S-DES is the function fk, which consists of a combination of permutation and substitution functions. The functions can be expressed as follows. Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input to f K, and let F be a mapping (not necessarily one to one) from 4-bit strings to 4-bit strings. Then we let

$$Fk (L, R) = (L \oplus F (R, SK), R)$$

Where SK is a sub key and $\oplus$ is the bit-by- bit exclusive OR function

Now, describe the mapping F. The input is a 4-bit number (n1 n2 n3 n4). The firstoperation is an expansion/permutation operation:

E/P							
4	1	2	3	2	3	4	1

Now, find the E/P from IP

IP = 01111110,

it becomes

E/P = 01111101

Now, XOR with K1

=> 01111101 $\oplus$ 10100100 = 11011001

The first 4 bits (first row of the preceding matrix) are fed into the S-box S0 to produce a 2- bit output, and the remaining 4 bits (second row) are fed into S1 to produce another 2-bit output.

These two boxes are defined as follows:

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

The S-boxes operate as follows. The first and fourth input bits are treated as a 2- bit number that specify a row of the S-box, and the second and third input bits specify a column of the S-box. Each s box gets 4-bit input and produce 2 bits as output. It follows 00- 0, 01-1, 10-2, 11-3 scheme.

Here, take first 4 bits, Second 4 bits

S0 => 1101 S1 => 1001

11 - > 3 11 -> 3

10 -> 2 => 3 =>11 00 -> 0 = > 2 => 10

So, we get 1110

Now, find P4

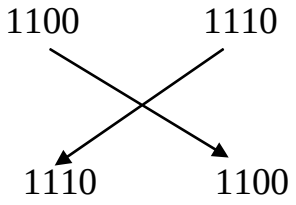
P4			
2	4	3	1

After P4, the value is 1011

Now, XOR operation 1011⊕0111 => 1100

3. The Switch function

The switch function (sw) interchanges the left and right 4 bits.



4. Second function fk

a. First, do E/P function and XOR with K2, the value is 01101001⊕01000011, the answer is00101010

b. Now, find S0 and S1

S_0 => 00 - > 0 01 -> 1 => 0 } S_1 = > 10 -> 2 01 -> 1 = > 0 => 00 = 00

Value is 0000

c. Now, find P4 and XOR operation

After P4 => 0000 ⊕1110 = 1110, then concatenate last 4 bits after interchange in sw.

Now value is 11101100

5. Find IP^{-1}

IP –1							
4	1	3	5	7	2	8	6

So, value is 01110101

The Ciphertext is 01110101

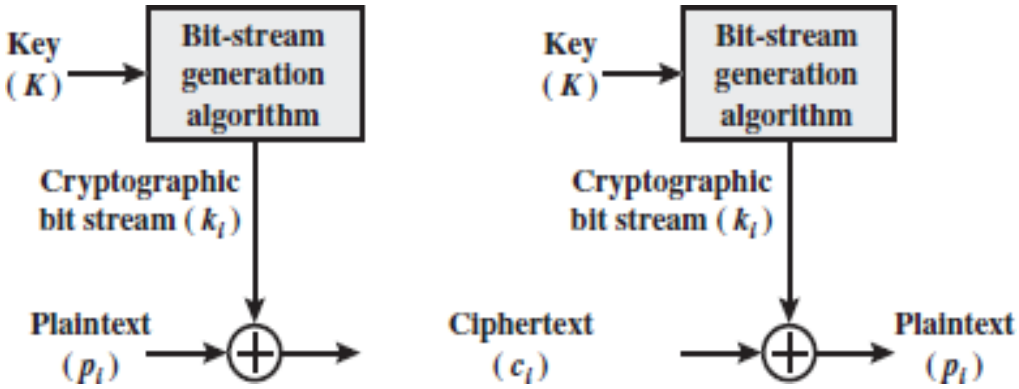
S-DES Decryption

- Decryption involves the sequential application of five functions.

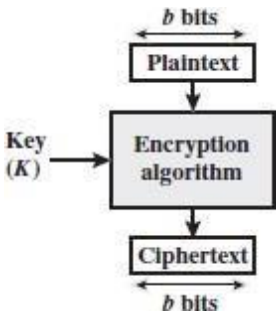
1. **Find IP**
   - After IP, value is 11101100
2. **Function f_k**
   - After step 2, the answer is 11101100
3. **Swift**
   - The answer is 11001110
4. **Second f_k**
   - The answer is 01111110
5. **Find IP-1**
   - **101111101 -> Plaintext**

2.9 BLOCK CIPHER

A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole andused to produce a ciphertext block of equal length.
A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenere cipher and the Vernam cipher.

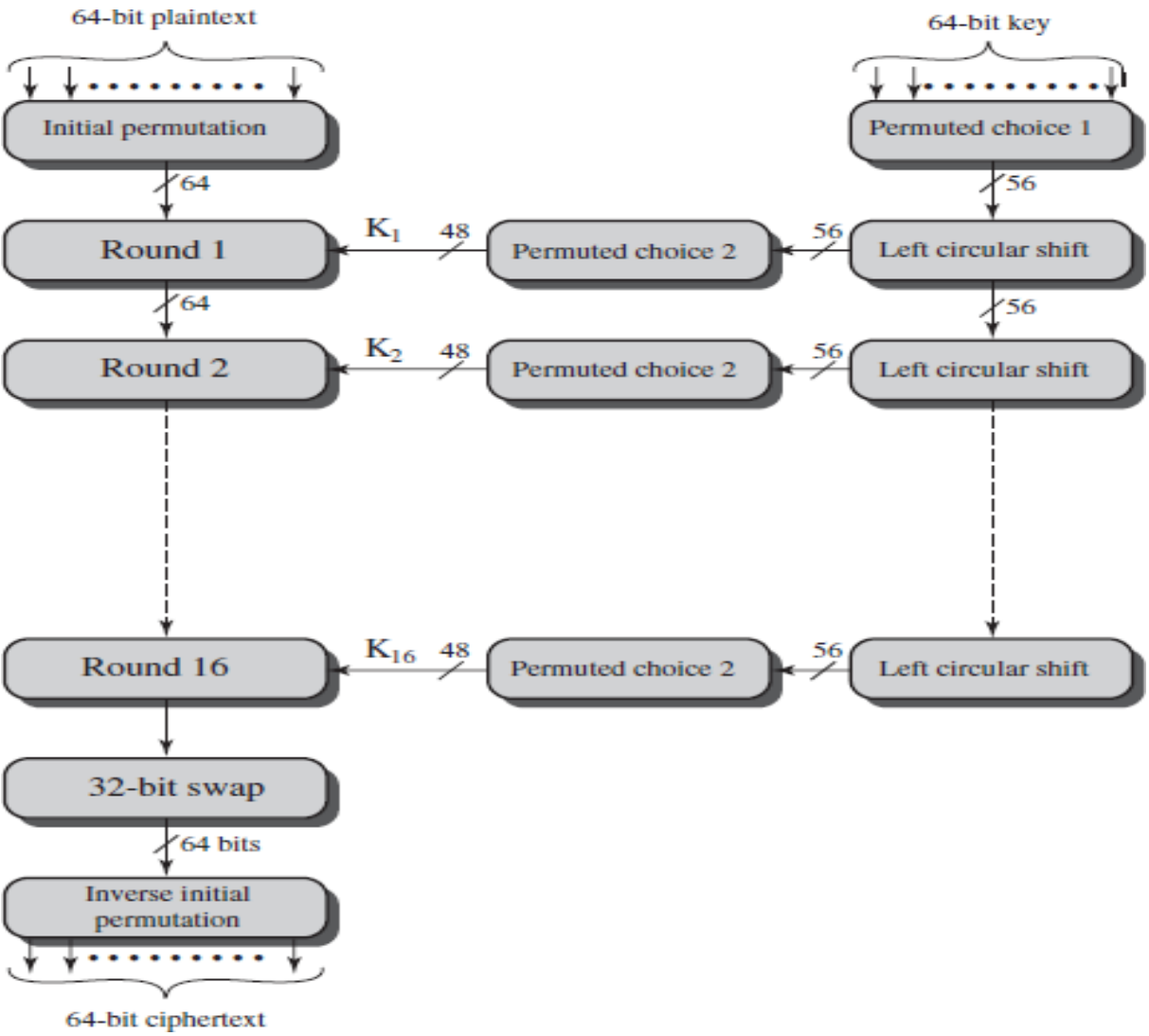


ENCRYPTION



DECRYPTION

2.10 DES - DATA ENCRYPTION STANDARD



Block Diagram of DES

As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.

Input of 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the **permuted input.** This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **pre output**.

Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text. 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a **subkey (Ki)** is produced by the combination of a left circular

shift and a permutation. The permutation function is the same for each round, but a different subkeyis produced because of the repeated shifts of the key bits.

(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation (IP^{-1})

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

Input PlainText M

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

Where M_i is a binary digit. Then the permutation X=IP(M)

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_{10}	M_2
M_{60}	M_{52}	M_{44}	M_{36}	M_{28}	M_{20}	M_{12}	M_4
M_{62}	M_{54}	M_{46}	M_{38}	M_{30}	M_{22}	M_{14}	M_6
M_{64}	M_{56}	M_{48}	M_{40}	M_{32}	M_{24}	M_{16}	M_8
M_{57}	M_{49}	M_{41}	M_{33}	M_{25}	M_{17}	M_9	M_1
M_{59}	M_{51}	M_{43}	M_{35}	M_{27}	M_{19}	M_{11}	M_3
M_{61}	M_{53}	M_{45}	M_{37}	M_{29}	M_{21}	M_{13}	M_5
M_{63}	M_{55}	M_{47}	M_{39}	M_{31}	M_{23}	M_{15}	M_7

The inverse permutation Y = IP-1(X) = IP-1(IP(M))

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities,labeled L (left) and R (right).

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

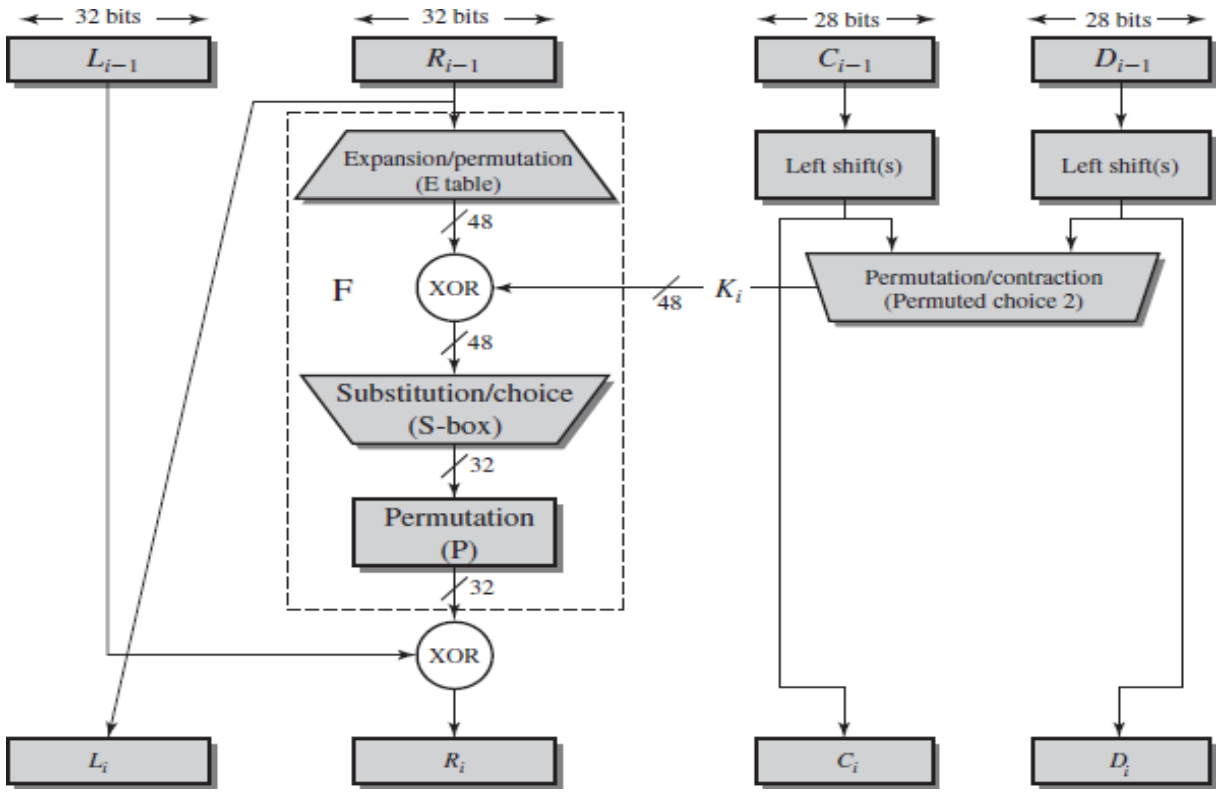
The round key Ki is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the Rbits. The resulting 48 bits are XORed with Ki.

This 48-bit result passes through a substitution function that produces a 32-bit output.

The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.

The first and last bits of the input to box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si.
The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output. For example,in S1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

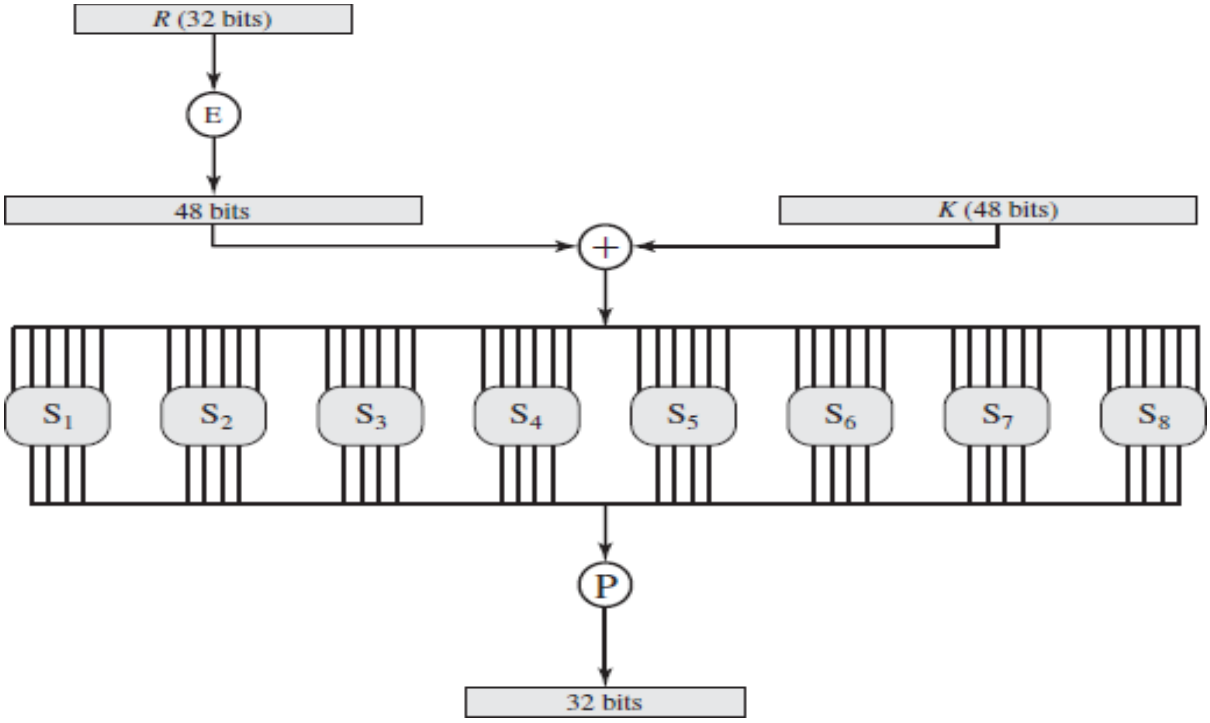
the input word is
... defghi hijklm lmnopq ...
this becomes **... efgh ijkl mnop ...**



Round Function

Key Generation

- 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64.
- The key is first subjected to a permutation governed by a table labeled Permuted Choice One. The resulting 56-bit key is then treated as two 28-bit quantities, labeled C0 and D0. At each round, Ci-1 and Di-1 are separately subjected toa circular left shift or (rotation) of 1 or 2 bits.
- These shifted values serve as input to the next round. They also serve as input to the part labeledPermuted Choice Two, which produces a 48-bit output that serves as input to the function F(Ri-1, Ki).



SUBSTITUTION BOX

DES Decryption

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application ofthe subkeys is reversed.

The Avalanche Effect

That a small change in either the plaintext or the key should produce a significant change in the cipher text. a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the cipher text. This is referred to as the avalanche effect.

(a) Input Key

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

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Substitution

Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertextelement or group of elements.

Permutation

A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

2.10 THE STRENGTH OF DES

The Use of 56-Bit Keys

With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 * 1016 keys.

The Nature of the DES Algorithm

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public.

There is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes. This assertion is tantalizing, and overthe years a number of regularities and unexpected behaviors of the S-boxes have been discovered.

Timing Attacks

A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various cipher texts. A timing attack exploits the fact that an encryption ordecryption algorithm often takes slightly different amounts of time on different inputs.

DES appears to be fairly resistant to a successful timing attack but suggest some avenues to explore. Although this is an interesting line of attack, it so far appears unlikely that this technique will ever be successful against DES or more powerful symmetric ciphers such as triple DES and AES.

2.11 DIFFERENTIAL AND LINEAR CRYPTANALYSIS

Differential Cryptanalysis

The differential cryptanalysis attack is complex; provides a complete description. The rationale behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block. The original plaintext block m to consist of two halves m0 and m1 . Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round. So, at each round, only one new 32-bit block is created. Intermediate message halves are related as

$$m_{i+1} = m_{i-1} \oplus f(m_i, K_i), \qquad i = 1, 2, \ldots, 16$$

In differential cryptanalysis, we start with two messages, m and m', with a known XOR difference $\Delta m = m \oplus m'$, and consider the difference between the intermediate message halves: $\Delta m_i = m_i \oplus m'_i$. Then we have

$$\begin{aligned} \Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [f(m_i, K_i) \oplus f(m'_i, K_i)] \end{aligned}$$

suppose that many pairs of inputs to f with the same difference yield the same output difference if the same subkey is used. if a number of such differences are determined, it is feasible to determine the subkey used in the function F.

Linear Cryptanalysis

This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given known plaintexts, as compared to chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES.

For a cipher with -bit plaintext and cipher text blocks and an -bit key, let the plaintext block be labelled P[1],....p[n], the cipher text block C[1],.....,C[n], and the key K[1], ,K[m] .Then define

$$A[i, j, \ldots, k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

The objective of linear cryptanalysis is to find an effective linear equation of the form

$$P[\alpha_1, \alpha_2, \ldots, \alpha_a] \oplus C[\beta_1, \beta_2, \ldots, \beta_b] = K[\gamma_1, \gamma_2, \ldots, \gamma_c]$$

2.12 BLOCK CIPHER DESIGN PRINCIPLES

DES Design Criteria

The criteria used in the design of DES, focused on the design of the S-boxes and on the P function that takes the output of the S-boxes .The criteria for the S-boxes are as follows.

1. No output bit of any S-box should be too close a linear function of the input bits. Specifically, if we select any output bit and any subset of the six input bits, the fraction of inputs for which this output bit equals the XOR of these input bits should not be close to 0 or 1, but rather should be near 1/2.

2. Each row of an S-box (determined by a fixed value of the leftmost and rightmost input bits) should include all 16 possible output bit combinations.

3. If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits.

4. If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.

5. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.

6. For any nonzero 6-bit difference between inputs, no more than eight of the 32 pairs of inputs exhibiting that difference may result in the same output difference.

7. This is a criterion similar to the previous one, but for the case of three S-boxes.

The criteria for the permutation P are as follows.

1. The four output bits from each S-box at round are distributed so that two of them affect (provide input for) “middle bits” of round (i+1) and the other two affect end bits. The two middle bits of input to an S-box are not shared with adjacent S-boxes. The end bits are the two left-hand bits and the two right-hand bits, which are shared with adjacent S-boxes.

2. The four output bits from each S-box affect six different S-boxes on the next round, and no two affect the same S-box.

3. For two S-boxes j,k if an output bit from Sj affects a middle bit of Sk on the next round, then an output bit from Sk cannot affect a middle bit of Sj . This implies that, for j=k , an output bit from Sj must not affect a middle bit of Sj.

Number of Rounds

The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F. In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack. The differential cryptanalysis attack requires 255.1 operations, whereas brute force 255 requires. If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than a brute-force key search.

Design of Function F

It must be difficult to "unscramble" the substitution performed by F. One obvious criterion is that F be nonlinear, as we discussed previously. The more nonlinear F, the more difficult any type of cryptanalysis will be. There are several measures of nonlinearity, which are beyond the scope of this book. In rough terms, the more difficult it is to approximate F by a set of linear equations, the more nonlinear F is.

S-Box Design

- Random: Use some pseudorandom number generation or some table of random digits to generate the entries in the S-boxes. This may lead to boxes with undesirable characteristics for small sizes but should be acceptable for large S-boxes.
- Random with testing: Choose S-box entries randomly, then test the results against various criteria, and throw away those that do not pass.
- Human-made: This is a more or less manual approach with only simple mathematics to support it. It is apparently the technique used in the DES design. This approach is difficult to carry through for large S-boxes.
- Math-made: Generate S-boxes according to mathematical principles. By using mathematical construction, S-boxes can be constructed that offer proven security against linear and differential cryptanalysis, together with good diffusion.

Key Schedule Algorithm

The key is used to generate one subkey for each round. In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key. The key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

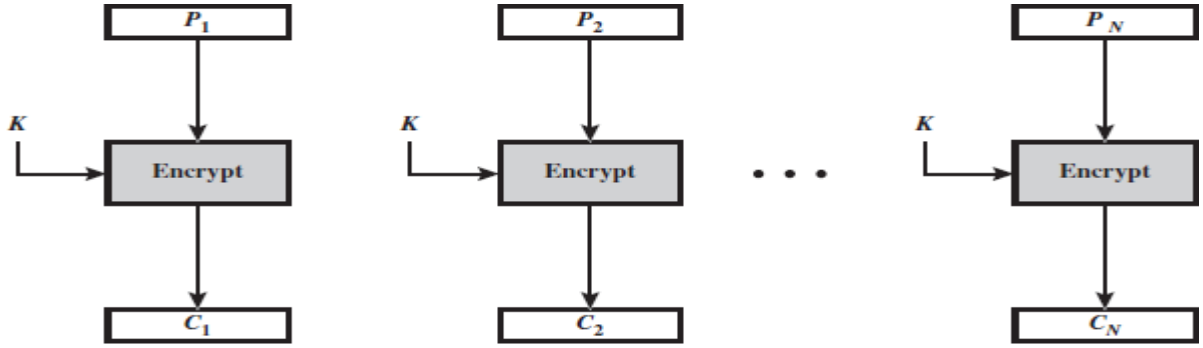
2.13 BLOCK CIPHER MODES OF OPERATION

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

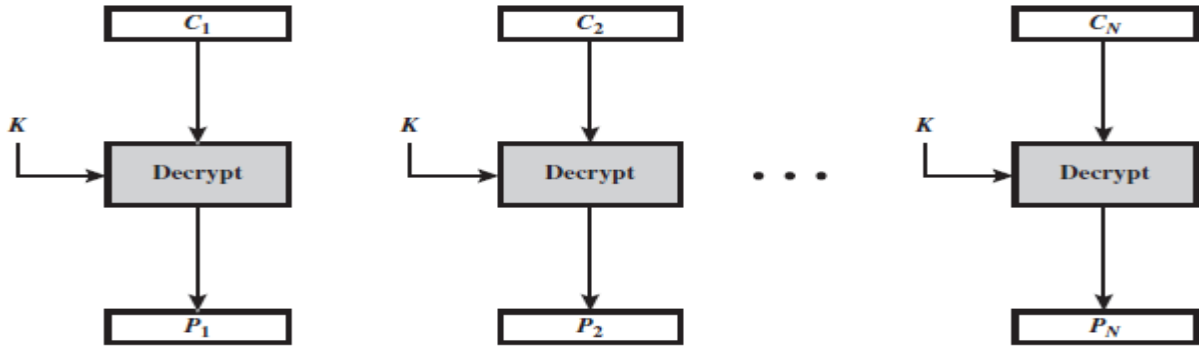
Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	Secure transmission of single values.
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	General-purpose block oriented Authentication
Cipher Feedback (CFB)	Input is processed bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	General-purpose stream oriented transmission Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.	Stream-oriented transmission over noisy channel
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	General-purpose block oriented Transmission Useful for high-speed requirements

ELECTRONIC CODEBOOK (ECB)

In which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key. The term **codebook** is used because, for a given key, there is a unique ciphertext for every -bit block of plaintext.



(a) Encryption



(b) Decryption

Cj = E(K, Pj) j = 1, ,N

Pj = D(K, Cj) j = 1, ,N

CIPHER BLOCK CHAINING MODE

The input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block.

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

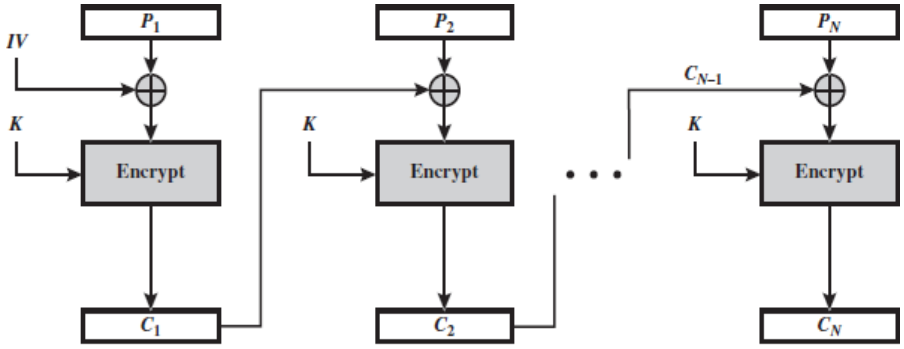
$$\mathrm{D}(K, C_j) = \mathrm{D}(K, E(K, [C_{j-1} \oplus P_j]))$$

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

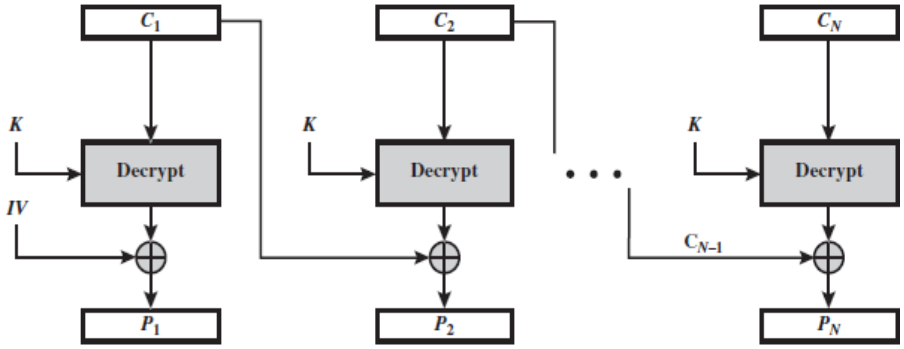
$$C_{j-1} \oplus \mathrm{D}(K, C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

$$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}]) \qquad P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$$

$$C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}]) \; j = 2, \ldots, N \qquad P_j = \mathrm{D}(K, C_j) \oplus C_{j-1} \; j = 2, \ldots, N$$



(a) Encryption



(b) Decryption

The IV must be known to both the sender and receiver but be unpredictable by a third party. In particular, for any given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV. For maximum security, the IV should be protected against unauthorized changes. This could be done by sending the IV using ECB encryption.

CIPHER FEEDBACK MODE

For AES, DES, or any block cipher, encryption is performed on a block of *b* bits. In the case of DES, *b* = 64 and in the case of AES, *b* = 128.

The CFB scheme, the unit of transmission is *s* bits; a common value is *s* = 8. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than blocks of b bits, the plaintext is divided into segments of s bits

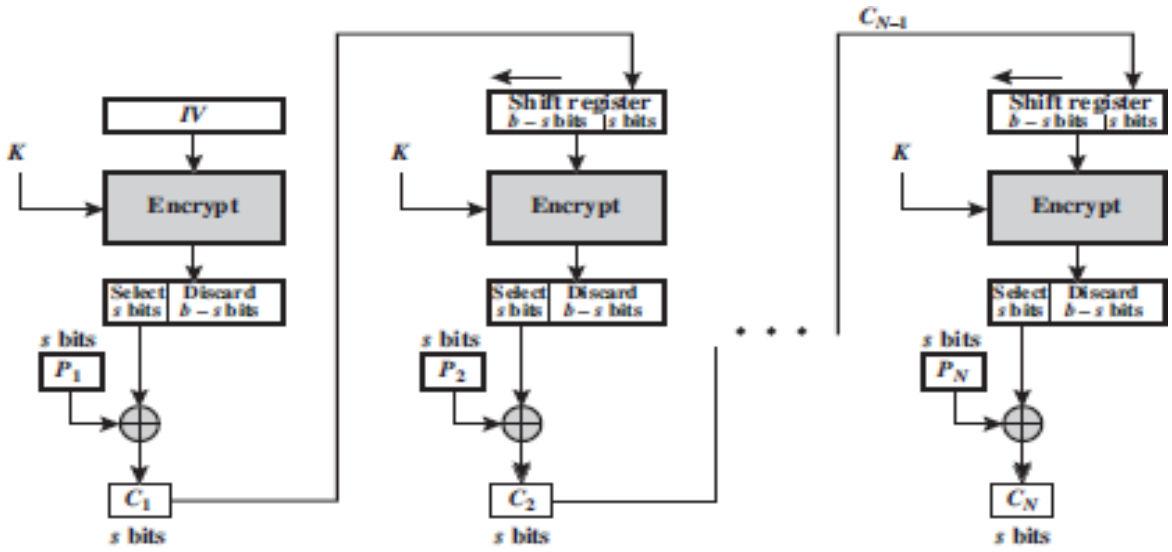
The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV).The leftmost (most significant) bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of ciphertext C2, which is then transmitted. In addition, the contents of the shift register are shifted left by bits, and C1 is placed in the rightmost (least significant) bits of the shift register. This process continues until all plaintext units have been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the encryption function that is used, not the decryption function.

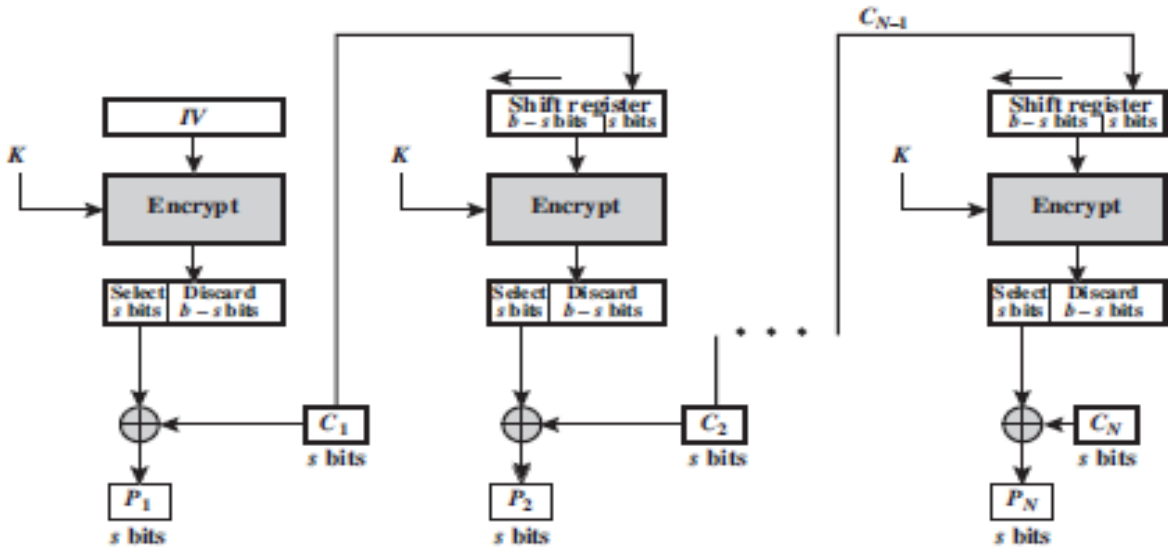
$$C_1 = P_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$
$$P_1 = C_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

$$\begin{aligned} I_1 &= IV \\ I_j &= \text{LSB}_{b-s}(I_{j-1}) \,\|\, C_{j-1} \quad j = 2, \ldots, N \\ O_j &= \text{E}(K, I_j) \quad j = 1, \ldots, N \\ C_j &= P_j \oplus \text{MSB}_s(O_j) \quad j = 1, \ldots, N \end{aligned}$$

$$\begin{aligned} I_1 &= IV \\ I_j &= \text{LSB}_{b-s}(I_{j-1}) \,\|\, C_{j-1} \quad j = 2, \ldots, N \\ O_j &= \text{E}(K, I_j) \quad j = 1, \ldots, N \\ P_j &= C_j \oplus \text{MSB}_s(O_j) \quad j = 1, \ldots, N \end{aligned}$$



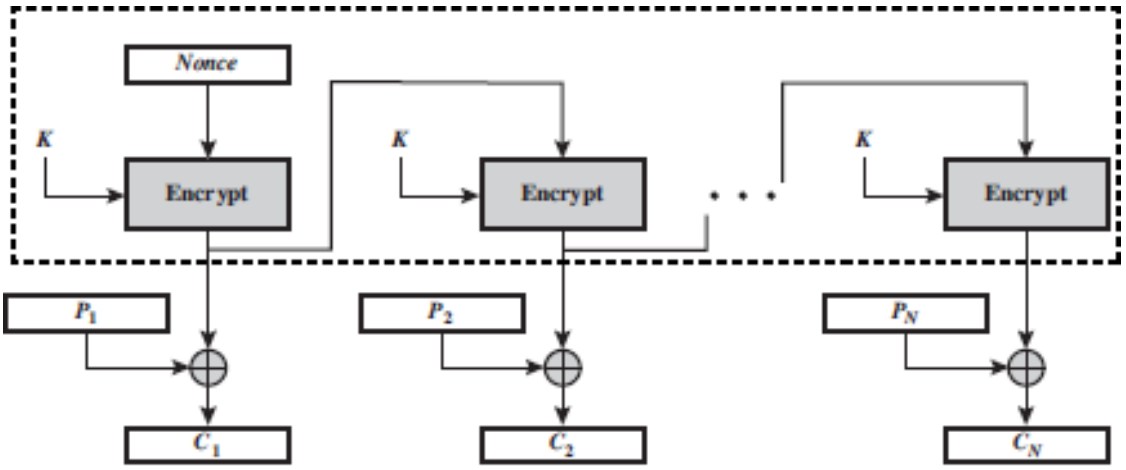
(a) Encryption



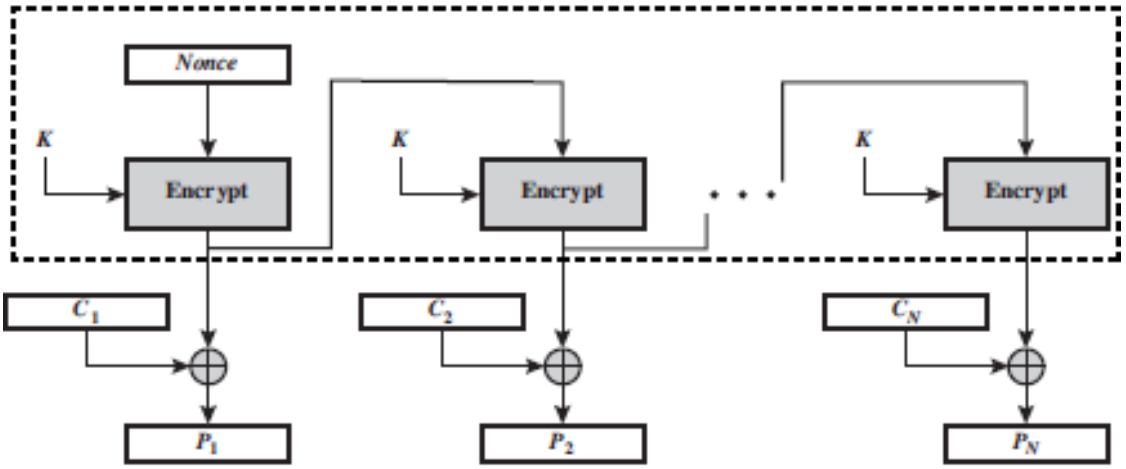
(b) Decryption

OUTPUT FEEDBACK MODE

The **output feedback** (OFB) mode is similar in structure to that of CFB. The output of the encryption function that is fed back to the shift register in OFB,whereas in CFB, the ciphertext unit is fed back to the shift register. The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, not on an b-bit subset.



(a) Encryption



(b) Decryption

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

$I_1 = Nonce$
$I_j = O_{j-1} \quad j = 2, \ldots, N$
$O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$
$C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$
$C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$

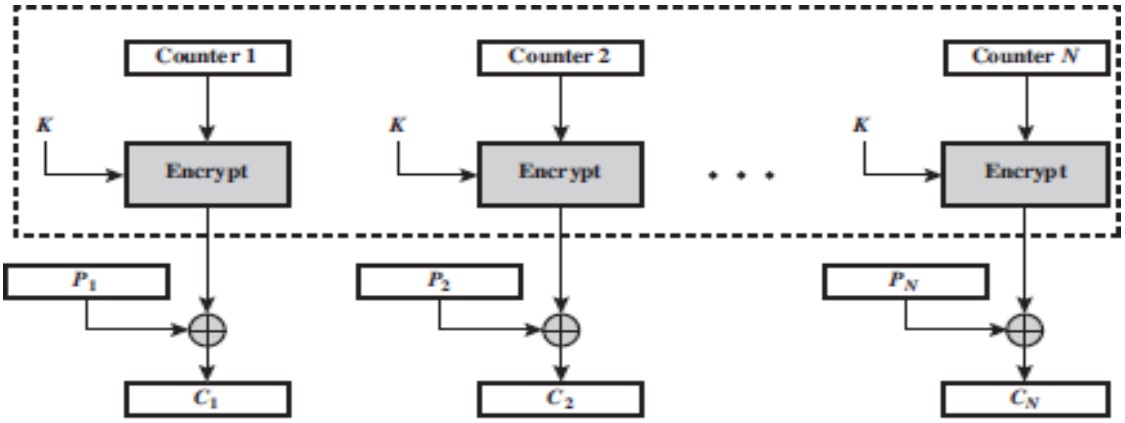
$I_1 = Nonce$
$I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \,\|\, C_{j-1} \quad j = 2, \ldots, N$
$O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$
$P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$
$P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$

COUNTER MODE

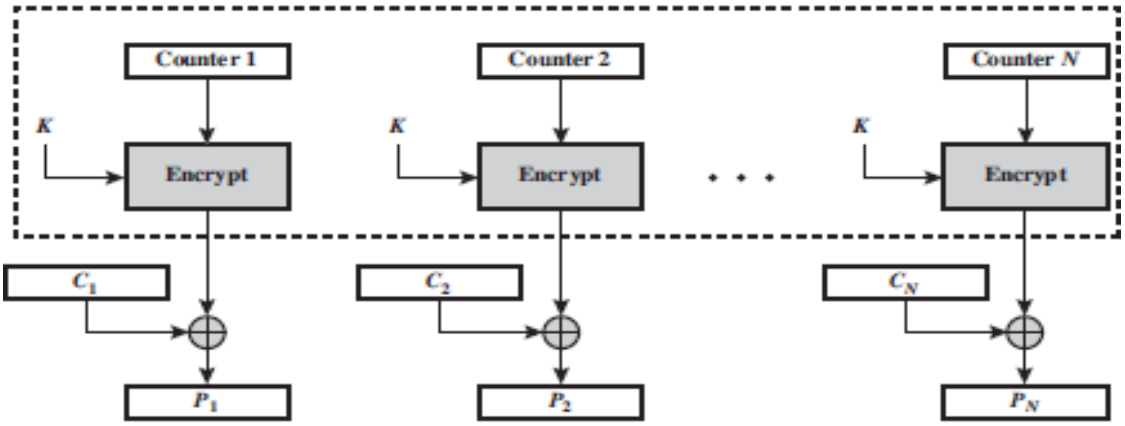
The **counter** (CTR) mode has increased recently with applications to ATM (asynchronous transfer mode) network security and IP sec (IP security). The counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^b where is the block size). For encryption, the counter 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. Thus, the initial counter value must be made available for decryption.

$$C_j = P_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$$
$$C_N^* = P_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$$

$$P_j = C_j \oplus \mathrm{E}(K, T_j) \quad j = 1, \ldots, N-1$$
$$P_N^* = C_N^* \oplus \mathrm{MSB}_u[\mathrm{E}(K, T_N)]$$



(a) Encryption



(b) Decryption

Advantages of CTR Mode:

Hardware efficiency: Unlike the three chaining modes, encryption (or decryption) in CTR mode can be done in parallel on multiple blocks of plaintext or ciphertext. For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block. This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption or decryption. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved.

Software efficiency: Similarly, because of the opportunities for parallel execution in CTR mode, processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions, can be effectively utilized.

Preprocessing: The execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes that feed into the XOR functions.

Random access: The th block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block Ci cannot be computed until the i-1 prior block are computed. There may be applications in which a ciphertext is stored and it is desired to decrypt just one block; for such applications, the random access feature is attractive.

Provable security: It can be shown that CTR is at least as secure as the othermodes discussed in this section.

Simplicity: Unlike ECB and CBC modes, CTR mode requires only the implementation of the encryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES. In addition, the decryption key schedulingneed not be implemented.

2.14 ADVANCED ENCRYPTION STANDARD

AES is a block cipher intended to replace DES for commercial applications. It uses a 128-bit blocksize and a key size of 128, 192, or 256 bits.Each full round consists of four separate functions: byte substitution, permutation, arithmeticoperations, over a finite field, and XOR with a key.

AES Parameters

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	10	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

Detailed Structure

1. AES instead processes the entire data block as a single matrix during each round usingsubstitutions and permutation.

2. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Fourdistinct words (128 bits) serve as a round key for each round.

3. Four different stages are used, one of permutation and three of substitution:
• Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
• ShiftRows: A simple permutation
• MixColumns: A substitution that makes use of arithmetic over
• AddRoundKey: A simple bitwise XOR of the current block
with a portion of the expanded key.

4. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages.
5. Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.

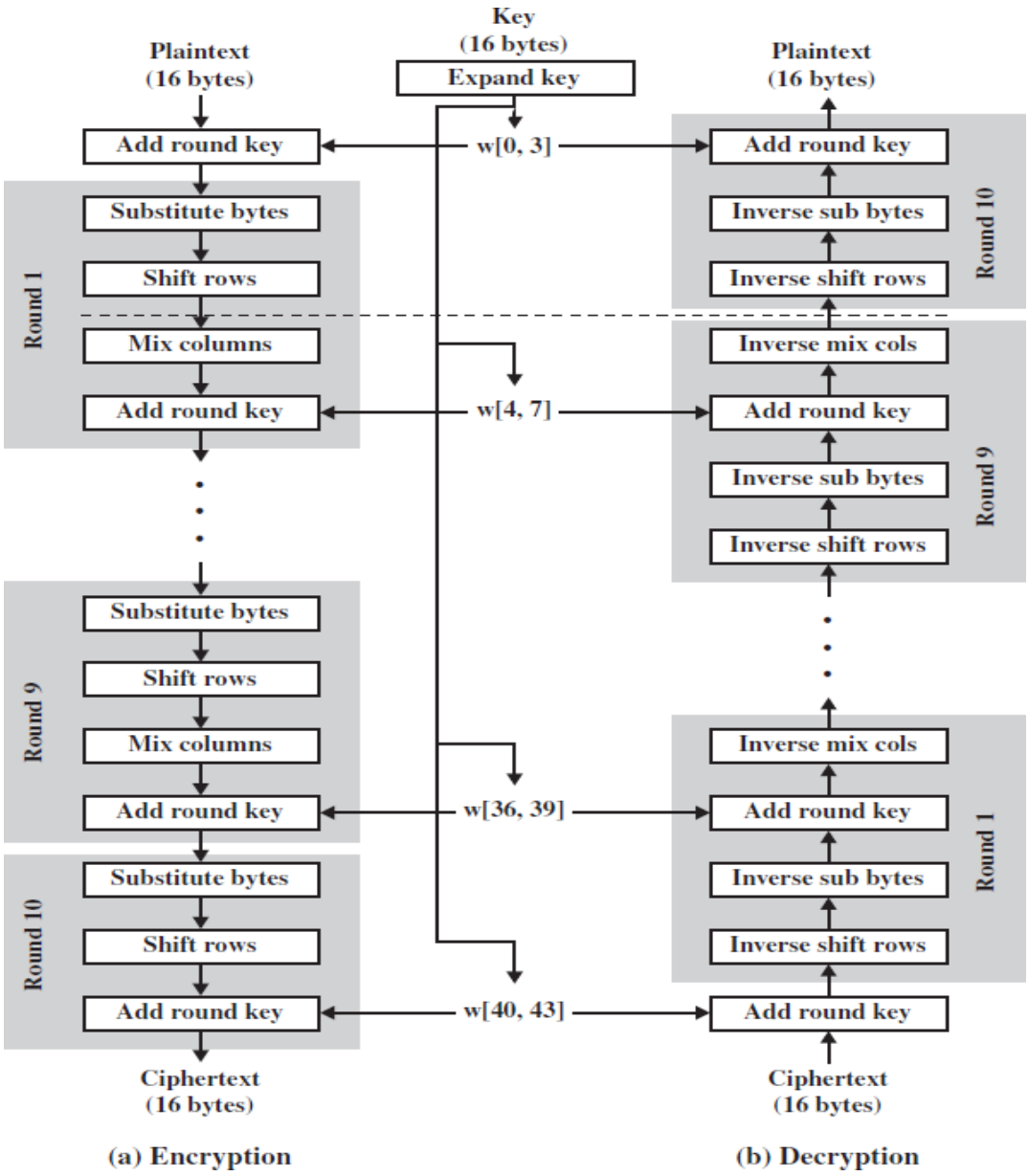
6. The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on.This scheme is both efficient and highly secure.

7. Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm.
For the AddRoundKey stage, the inverse is achieved by XORing the same round key to the block.

$$A \oplus B \oplus B = A.$$

8. The decryption algorithm makes use of the expanded key in reverse order. The decryption algorithm is not identical to the encryption algorithm.

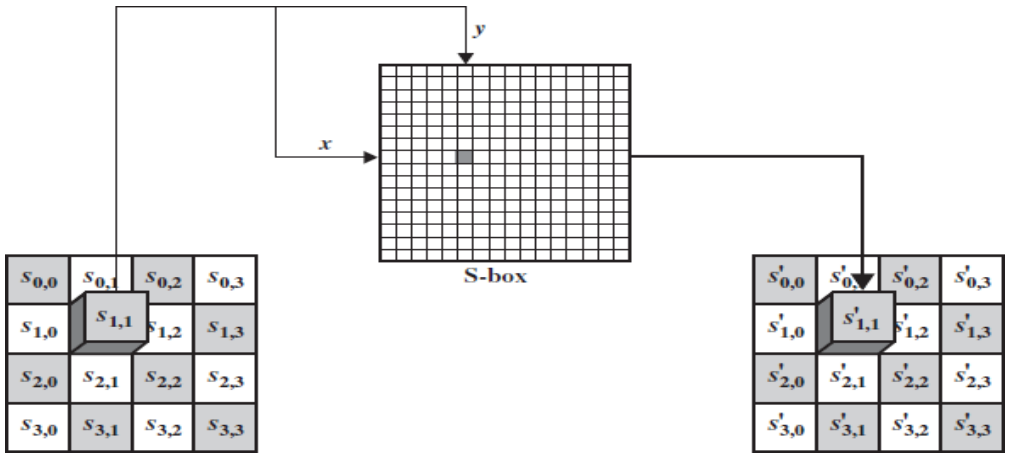
9. The final round of both encryption and decryption consists of only three Stages. This is a consequence of the particular structure of aes and is required to make the cipher reversible.



BLOCK DIAGRAM OF AES

Substitute Bytes Transformation

The forward substitute byte transformation, called SubBytes, is a simple table lookup. AES defines a 16X16 matrix of byte values, called an S-box. That contains a permutation of all possible 256 8-bit values. Each individual byte of State is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value. For example, the hexadecimal value {95} references row 9,column 5



EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

SUBSTITUTION BYTE TRANSFORMAION

S BOX

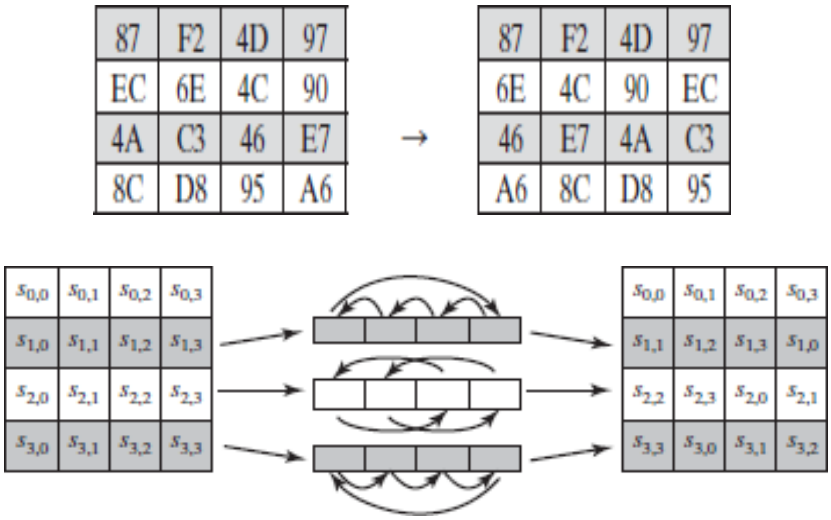
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	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

INVERSE BOX

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

ShiftRows Transformation

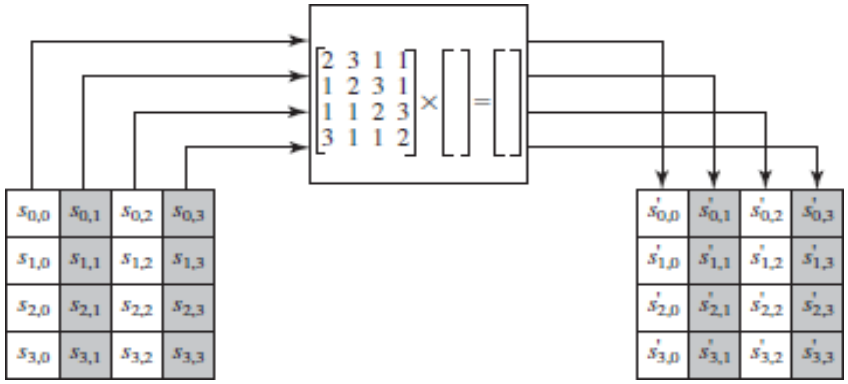
The first row of State is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the fourth row, a 3-byte circular left shift is performed. The inverse shift row transformation, called InvShiftRows, performs the circular shifts in the opposite direction for each of the last three rows, with a 1-byte circular right shift for the second row, and so on.



SHIFT ROW TRANSFORMATION

MixColumn Transformation

Each byte of a column is mapped into a new value that is a function of all four bytes in that column. Multiplication of a value by (i.e., by {02}) can be implemented as a 1-bit left shiftfollowed by a conditional bitwise XOR with (0001 1011) if the leftmost bit of the original value(prior to the shift) is 1. Thus, to verify the MixColumns transformation on the first column



$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

$$\begin{aligned} s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j} \\ s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \\ s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j}) \\ s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j}) \end{aligned}$$

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$$\begin{aligned} (\{02\} \cdot \{87\}) \oplus (\{03\} \cdot \{6E\}) \oplus \{46\} \oplus \{A6\} &= \{47\} \\ \{87\} \oplus (\{02\} \cdot \{6E\}) \oplus (\{03\} \cdot \{46\}) \oplus \{A6\} &= \{37\} \\ \{87\} \oplus \{6E\} \oplus (\{02\} \cdot \{46\}) \oplus (\{03\} \cdot \{A6\}) &= \{94\} \\ (\{03\} \cdot \{87\}) \oplus \{6E\} \oplus \{46\} \oplus (\{02\} \cdot \{A6\}) &= \{ED\} \end{aligned}$$

$$\begin{aligned} \{02\} \cdot \{87\} &= 0001\ 0101 \\ \{03\} \cdot \{6E\} &= 1011\ 0010 \\ \{46\} &= 0100\ 0110 \\ \{A6\} &= \underline{1010\ 0110} \\ & 0100\ 0111 = \{47\} \end{aligned}$$

AddRoundKey Transformation

The 128 bits of State are bitwise XORed with the 128 bits of the round key. The operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key; it can also be viewed as a byte-level operation

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

AES Key Expansion

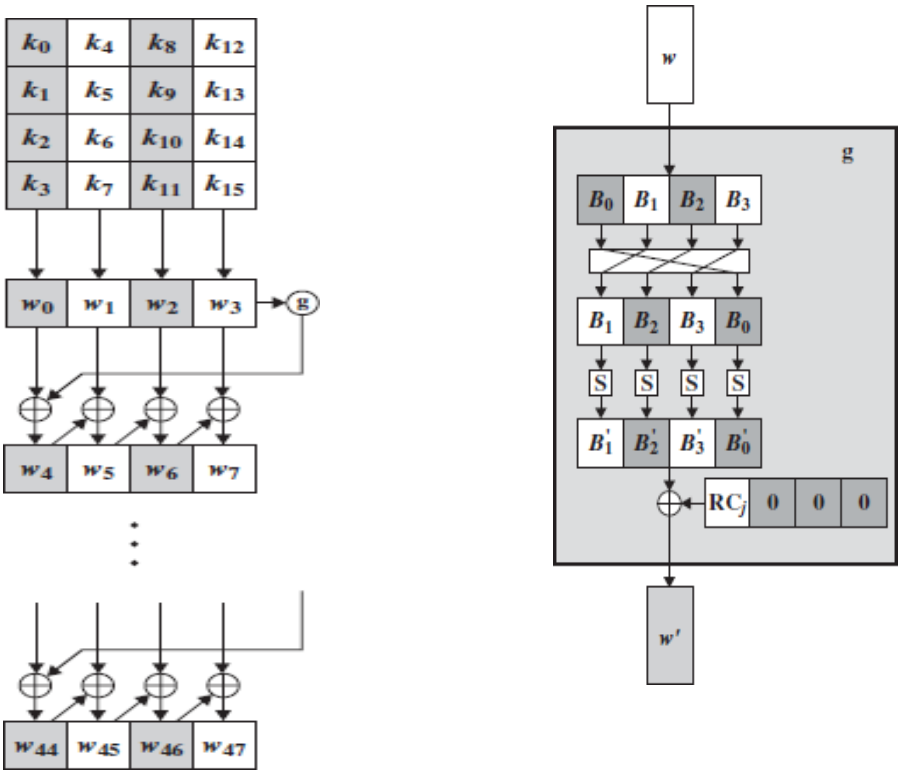
The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes).This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.

```
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 XOR
(RotWord (temp))Rcon[i/4];
w[i] = w[i–4] XOR temp
}
}
```

1. RotWord performs a one-byte circular left shift on a word.This means that an input word [B0,B1, B2, B3] is transformed into [B1, B2, B3, B0].

2. SubWord performs a byte substitution on each byte of its input word, using the S-box.

3. The result of steps 1 and 2 is XORed with a round constant Rcon[j]



j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

AES KEY EXPANSION

2.15 RC4 ENCRYPTION ALGORITHM

RC4 is a stream cipher and variable length key algorithm. This algorithm encrypts one byte at a time (or larger units on a time).

A key input is pseudorandom bit generator that produces a stream 8-bit number that is unpredictable without knowledge of input key, The output of the generator is called key-stream, is combined one byte at a time with the plaintext stream cipher using X-OR operation.

Example:

```
RC4 Encryption

10011000 ? 01010000 = 11001000

RC4 Decryption
11001000 ? 01010000 = 10011000
```

2.15.1 Key-Generation Algorithm –

A variable-length key from 1 to 256 byte is used to initialize a 256-byte state vector S, with elements S[0] to S[255]. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion, then the entries in S are permuted again.

Key-Scheduling Algorithm:

Initialization: The entries of S are set equal to the values from 0 to 255 in ascending order, a temporary vector T, is created.

If the length of the key k is 256 bytes, then k is assigned to T. Otherwise, for a key with length(k-len) bytes, the first k-len elements of T as copied from K and then K is repeated as many times asnecessary to fill T. The idea is illustrated as follow:

```
for
    i = 0 to 255 do S[i] = i;
T[i]= K[i mod k - len];
```

we use T to produce the initial permutation of S. Starting with S[0] to S[255], and for each S[i] algorithm swap it with another byte in S according to a scheme dictated by T[i], but S will still contain values from 0 to 255 :

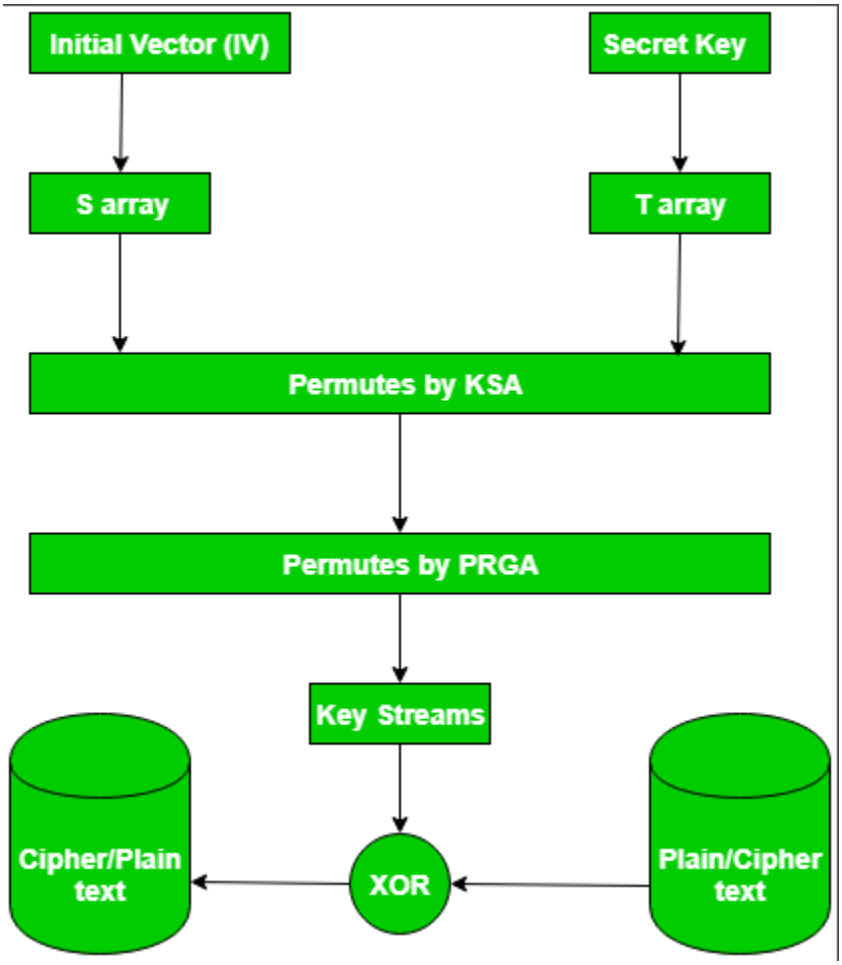
```
j = 0;
for
    i = 0 to 255 do
    {
        j = (j + S[i] + T[i])mod 256;
        Swap(S[i], S[j]);
    }
```

2.15.2 Pseudo random generation algorithm (Stream Generation):

Once the vector S is initialized, the input key will not be used. In this step, for each S[i] algorithm swap it with another byte in S according to a scheme dictated by the current configuration of S.
After reaching S[255] the process continues, starting from S[0] again

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

2.15.3 Encrypt using X-Or()



RC4 Algorithm

In the RC4 encryption algorithm, the key stream is completely independent of the plaintext used. An 8 * 8 S-Box (S0 S255), where each of the entries is a permutation of the numbers 0 to 255, and the permutation is a function of the variable length key. There are two counters i, and j, bothinitialized to 0 used in the algorithm.

The algorithm uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream which is XORed with the plaintext to give the ciphertext. Each element in the state table is swapped at least once.

The key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits. RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure SQL.

The algorithm works in two phases, key setup and ciphering. Key setup is the first and most difficult phase of this encryption algorithm. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulas. A modulo operation is the process of yielding a remainder from division. For example, 11/4 is 2 remainder 3; therefore eleven mod four would be equal to three.

Strengths of RC4

The difficulty of knowing where any value is in the table.

The difficulty of knowing which location in the table is used to select each value in the sequence.Encryption is about 10 times faster than DES.

Limitations of RC4

RC4 is no longer considered secure.

One in every 256 keys can be a weak key. These keys are identified by cryptanalysis that is able to find circumstances under which one of more generated bytes are strongly correlated with a few bytes of the key.

A particular RC4 Algorithm key can be used only once.

2.16 KEY DISTRIBUTION

Discussed briefly in Unit 3