CHAPTER 2

SYMMETRIC CIPHERS

CHAPTER OUTLINE

After studying this chapter, the students will be able to understand the

- Fiestel Cipher Structure, Substitution Permutation Network (SPN)
- Data Encryption Standards (DES), Double DES, Triple DES
- Finite Fields: Groups Rings, Fields, Modular Arithmetic, Euclidean Algorithm, Galois Fields (GF(p) & GF(2n)), Polynomial Arithmetic
- International Data Encryption Standard (IDEA)
- Advanced Encryption Standards (AES) Cipher
- Modes of Block Cipher Encryptions (Electronic Code Book, Cipher Block Chaining, Cipher Feedback Mode, Output Feedback Mode, Counter Mode)

THE FEISTEL CIPHER

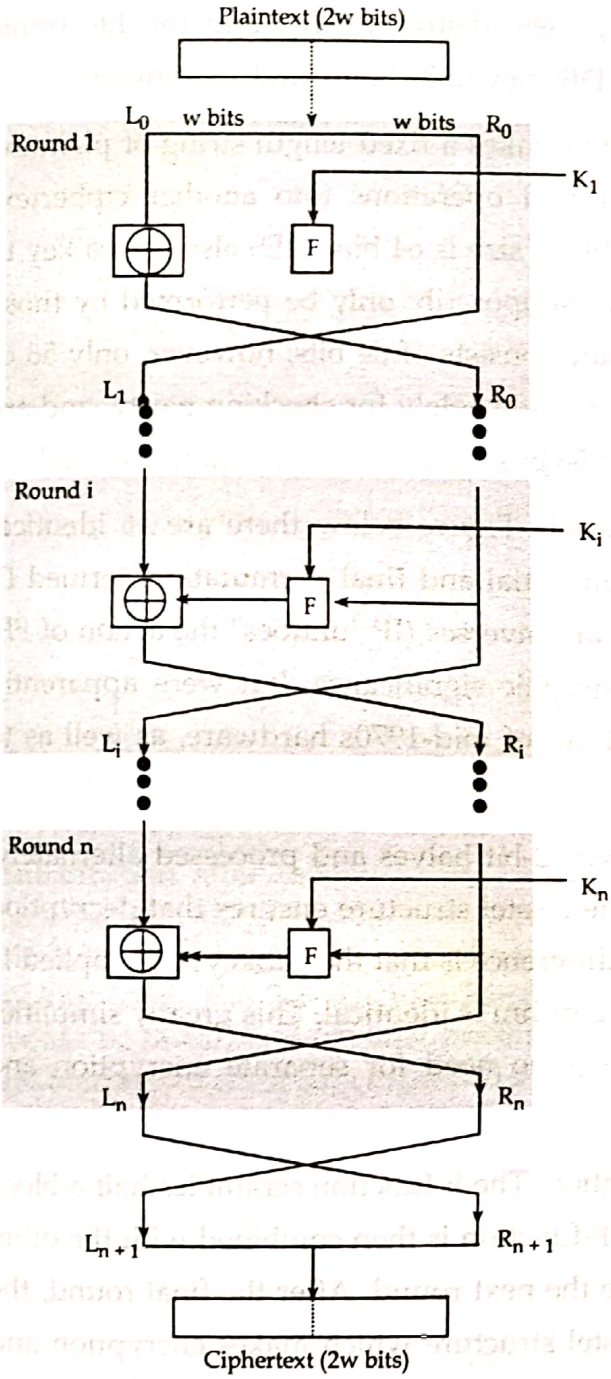
Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of n bits, allowing a total of 2k possible transformations, rather than the 2n! transformations available with the ideal block cipher.

DIFFUSION AND CONFUSION

The terms diffusion and confusion were introduced by Claude Shannon to capture the two basic building blocks for any cryptographic system. Shannon's concern was to thwart cryptanalysis based on statistical analysis. The reasoning is as follows. Assume the attacker has some knowledge of the statistical characteristics of the plaintext. For example, in a human-readable message in some language, the frequency distribution of the various letters may be known. Or there may be words or phrases likely to appear in the message (probable words). If these statistics are in any way reflected in the ciphertext, the cryptanalyst may be able to deduce the encryption key, or part of the key, or at least a set of keys likely to contain the exact key. In what Shannon refers to as a strongly ideal cipher, all statistics of the ciphertext are independent of the particular key used.

FEISTEL CIPHER STRUCTURE

Figure depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into two halves, L0 and R_0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs Li-1 and Ri-1, derived from the previous round, as well as a subkey Ki, derived from the overall K. In general, the subkeys Ki are different from K and from each other.



All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey Ki. Following this substitution, a **permutation** is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

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

Block size, Key size, Number of rounds, Subkey generation algorithm, Round function, Fast software encryption/decryption, Ease of analysis.

Figure 2.1: Classical Feistel Network

DATA ENCRYPTION STANDARD (DES)

The *Data Encryption Standard (DES)* is a cipher (a method for encrypting information) selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976, and which has subsequently enjoyed widespread use internationally. The algorithm was initially controversial, with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES consequently came under intense academic scrutiny, and motivated the modern understanding of block ciphers and their cryptanalysis. DES is now considered to be insecure for many applications. This is chiefly due

to the 56-bit key size being too small; in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes.

Description: DES is the block cipher - an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits, and it is usually quoted as such.

Structure: The algorithm's overall structure is shown in Figure below there are 16 identical stages of processing, termed rounds. There is also an initial and final permutation, termed IP and FP (see appendix for IP and FP scheme), which are inverses (IP "undoes" the action of FP, and vice versa). IP and FP have almost no cryptographic significance, but were apparently included in order to facilitate loading blocks in and out of mid-1970s hardware, as well as to make DES run slower in software.

Before the main rounds, the block is divided into two 32-bit halves and processed alternately; this criss-crossing is known as the Feistel scheme. The Feistel structure ensures that decryption and encryption are very similar processes - the only difference is that the subkeys are applied in the reverse order when decrypting. The rest of the algorithm is identical. This greatly simplifies implementation, particularly in hardware, as there is no need for separate encryption and decryption algorithms.

The ⊕ symbol denotes the exclusive-OR (XOR) operation. The F-function scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

The Feistel (F) function (Mangler Function): The F-function, depicted in Figure below, operates on half a block (32 bits) at a time and consists of four stages:

Expansion: the 32-bit half-block is expanded to 48 bits using the expansion permutation (see appendix for expansion permutation), denoted E in the diagram, by duplicating some of the bits.

Key Mixing: the result is combined with a subkey using an XOR operation. Sixteen 48-bit subkeys - one for each round - are derived from the main key using the key schedule described below.

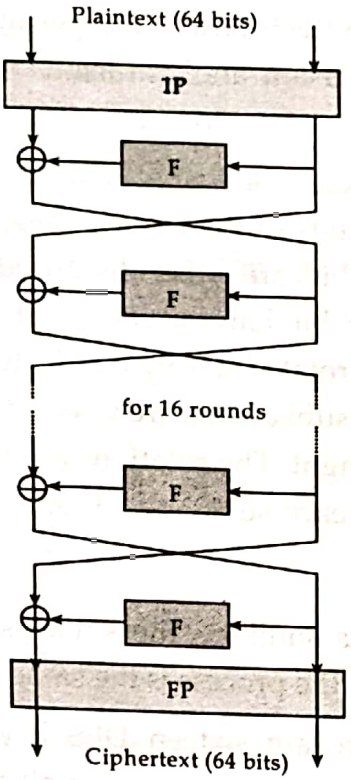


Figure 2.2: DES Encryption

Substitution after mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the S-boxes, or substitution boxes. Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, provided in the form of a lookup table. The S-boxes provide the core of the security of DES - without them, the cipher would be linear, and trivially breakable. Permutation: finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation, the P-box.

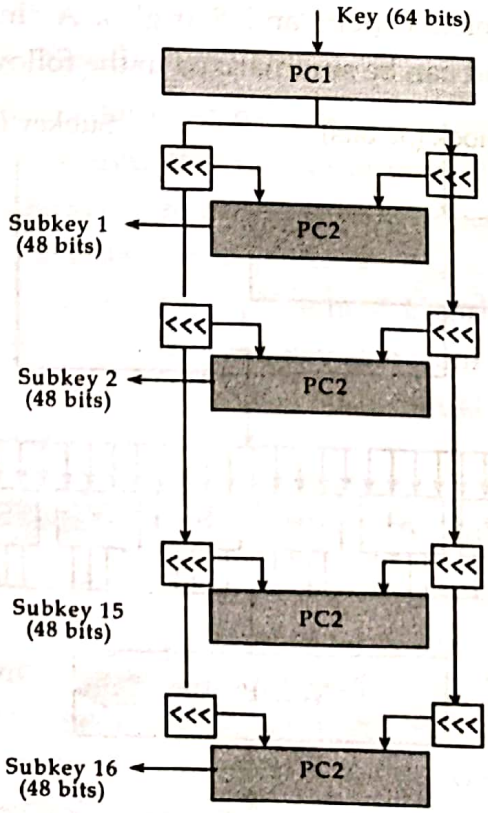


Figure 2.3: Key Generation (16 Rounds)

The alternation of substitution from the S-boxes, and permutation of bits from the P-box and E-expansion provides so-called "confusion and diffusion" respectively, a concept identified by Claude Shannon in the 1940s as a necessary condition for a secure yet practical cipher.

Key Schedule: Figure below illustrates the key schedule for encryption - the algorithm which generates the subkeys. Initially, 56 bits of the key are selected from the initial 64 by Permuted Choice 1, PC-1 the remaining eight bits are either discarded or used as parity check bits. The 56 bits are then divided into two 28-bit halves; each half is thereafter treated separately. In successive rounds, both halves are rotated left by one or two bits specified for each round (for key rotation schedule), and then 48 subkey bits are selected by Permuted Choice 2, PC-2, 24 bits from the left half, and 24 from the right. The rotations (denoted by "<<<" in the diagram) mean that a different set of bits is used in each subkey; each bit is used in approximately 14 out of the 16 subkeys.

The key schedule for decryption is similar - the subkeys are in reverse order compared to encryption. Apart from that change, the process is the same as for encryption.

Weak and Semi-Weak Keys: There are sixteen DES keys that are not suggested for use. However the probability of getting such keys is very small as given by $16/2^{56}$. The sixteen weak keys are keys with all ones, all zeroes, alternating zeroes and ones, and alternating ones and zeroes for two 28 bits parts of the key generated when PC-1 is used.

DETAILS OF SINGLE ROUND

Figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

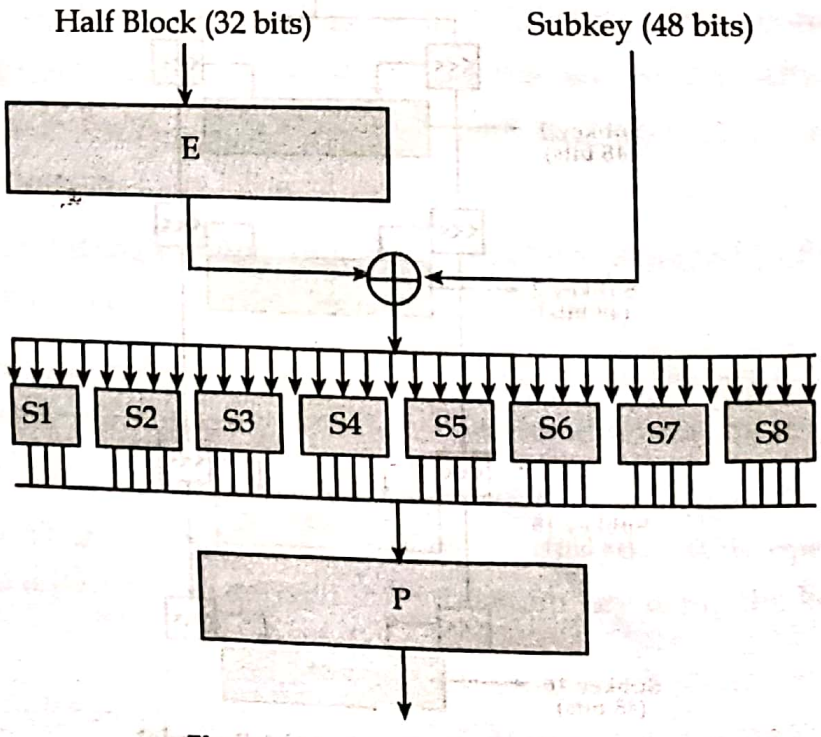


Figure 2.4: Calculation of F(R, K)

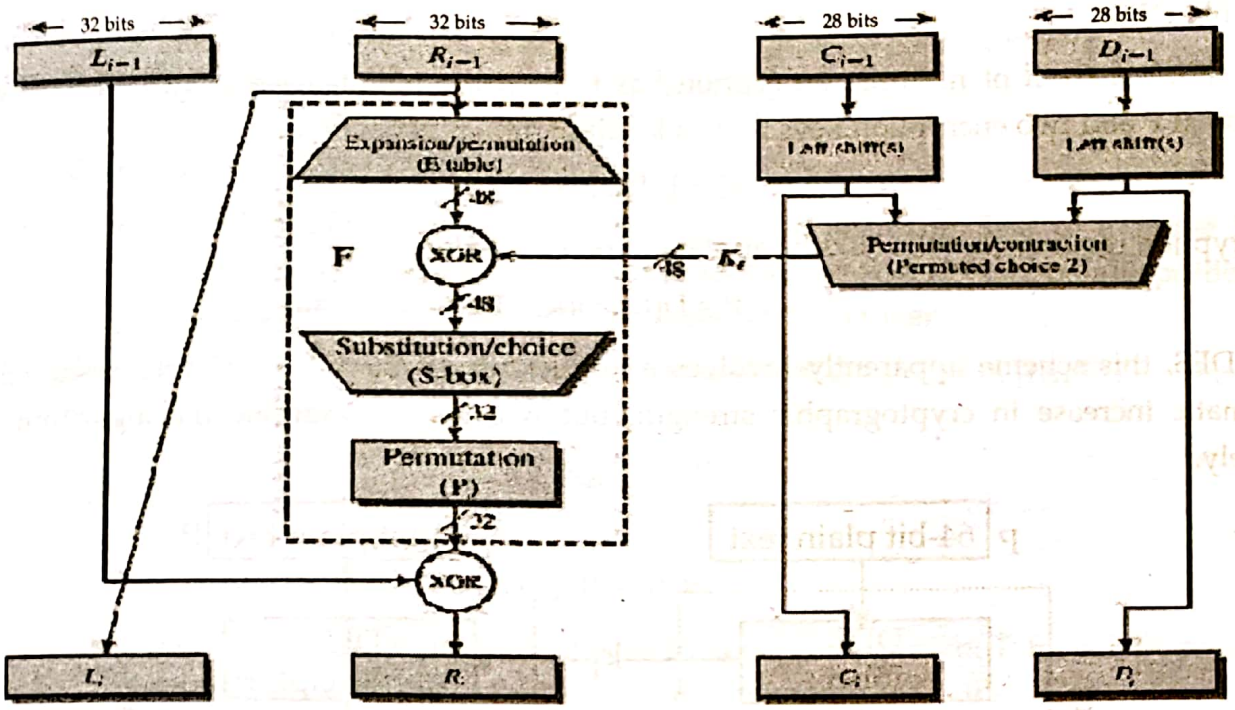


Figure 2.5: Single Round of DES Algorithm

SECURITY AND CRYPTANALYSIS OF DES

Although more information has been published on the cryptanalysis of DES than any other block cipher, the most practical attack to date is still a brute force approach. Various minor cryptanalytic properties are known, and three theoretical attacks are possible which, while having a theoretical complexity less than a brute force attack, require an unrealistic amount of known or chosen plaintext to carry out, and are not a concern in practice. In spite of all the criticism and weaknesses of DES, there is no known example of anyone actually suffering monetary losses because of DES security limitations.

Brute force attack: For any cipher, the most basic way of attack is brute force - trying every possible key. The length of the key gives the number of possible keys, and hence the feasibility of this approach. For DES, questions were raised about the adequacy of its key size early on, even before it was adopted as a standard, and it was the small key size, rather than theoretical cryptanalysis, which dictated a need for a replacement algorithm. It is known that the NSA encouraged, if not persuaded, IBM to reduce the key size from 128 to 64 bits, and from there to 56 bits; this is often taken as an indication that the NSA thought it would be able to break keys of this length even in the mid-1970s.

MULTIPLE ENCRYPTION AND TRIPLE DES

Given the potential vulnerability of DES to a brute-force attack, there has been considerable interest in finding an alternative. One approach is to design a completely new algorithm, of which AES is a prime example. Another alternative, which would preserve the existing investment in software and equipment, is to use multiple encryption with DES and multiple keys. We begin by examining the simplest example of this second alternative. We then look at the widely accepted triple DES (3DES) approach.

Double DES

The simplest form of multiple encryption has two encryption stages and two keys. Given a plaintext P and two encryption keys K_1 and k_2 ,ciphertext is generated as

$$C = E(K_2, E(K_1, P))$$

Decryption requires that the keys be applied in reverse order:

$$P = D(K_1, D(K_2, C))$$

For DES, this scheme apparently involves a key length of $56 \times 2 = 112$ bits, resulting in a dramatic increase in cryptographic strength. But we need to examine the algorithm more closely.

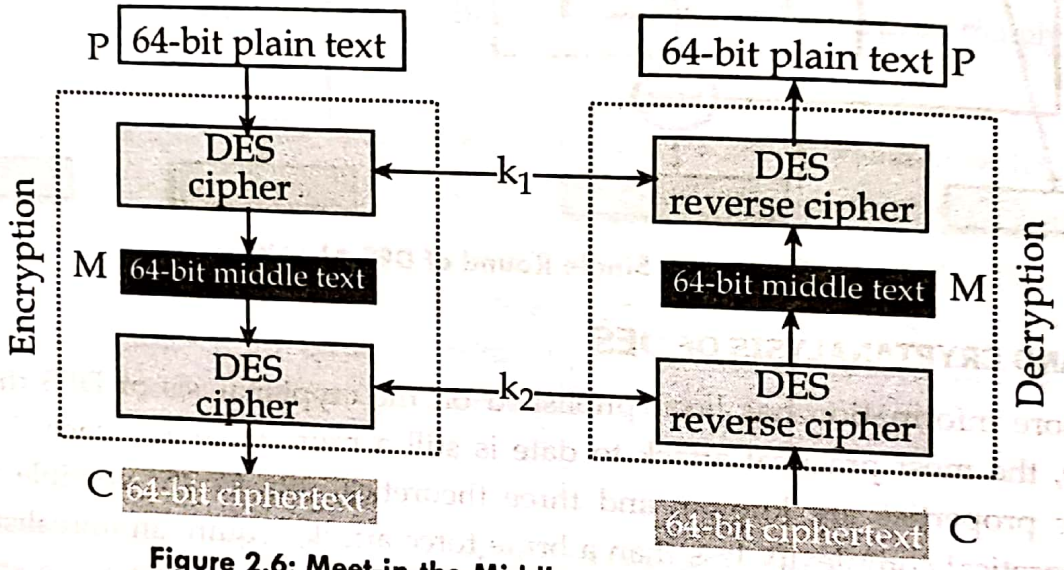


Figure 2.6: Meet-in-the-Middle attack for Double DES

REDUCTION TO A SINGLE STAGE

Suppose it were true for DES, for all 56-bit key values, that given any two keys k_1 and k_2 ,it would be possible to find a key k_3 such that

$E(K_2, E(K_1, P)) = E(K_3, P)$

If this were the case, then double encryption, and indeed any number of stages of multiple encryption with DES, would be useless because the result would be equivalent to a single encryption with a single 56-bit key.

Consider that encryption with DES is a mapping of 64-bit blocks to 64-bit blocks. In fact, the mapping can be viewed as a permutation. That is, if we consider 2^{64} all possible input blocks, DES encryption with a specific key will map each block into a unique 64-bit block. Otherwise, if, say, two given input blocks mapped to the same output block, then decryption to recover the original plaintext would be impossible. With 2^{64} possible inputs, how many different mappings are there that generate a permutation of the input blocks? The value is easily seen to be

$(264)! = 10347380000000000000000 > (10^{10^{20}})$

On the other hand, DES defines one mapping for each different key, for a total number of mappings:

$2^{56} < 10^{17}$

Therefore, it is reasonable to assume that if DES is used twice with different keys, it will produce one of the many mappings that are not defined by a single application of DES.

MEET-IN-THE-MIDDLE ATTACK

Thus, the use of double DES results in a mapping that is not equivalent to a single DES encryption. But there is a way to attack this scheme, one that does not depend on any particular property of DES but that will work against any block encryption cipher.

The algorithm, known as a meet-in-the-middle attack, was first described in [DIFF77].It is based on the observation that, if we have

$$C = E(K_2, E(K_1, P))$$

Then

$$X = E(K_1, P) = D(K_2, C)$$

Given a known pair, (P, C) ,the attack proceeds as follows. First, encrypt P for all 2^{56} possible values of K_1. Store these results in a table and then sort the table by the values of X. Next, decrypt C using all 2^{56} possible values of K_2. As each decryption is produced, check the result against the table for a match. If a match occurs, then test the two resulting keys against a new known plaintext–ciphertext pair. If the two keys produce the correct ciphertext, accept them as the correct keys.

For any given plaintext P, there are 2^{64} possible ciphertext values that could be produced by double DES. Double DES uses, in effect, a 112-bit key, so that there are 2^{112} possible keys. Therefore, on average, for a given plaintext P , the number of different 112-bit keys that will produce a given ciphertext is $2^{112}/2^{64} = 2^{48}$. Thus, the foregoing procedure will produce about false alarms on the first(P, C) pair .A similar argument indicates that with an additional 64 bits of known plaintext and ciphertext, the false alarm rate is reduced to $2^{48-64} = 2^{-16}$. Put another way, if the meet-in-the-middle attack is performed on two blocks of known plaintext–ciphertext, the probability that the correct keys are determined is $1 - 2^{-16}$. The result is that a known plaintext attack will succeed against double DES, which has a key size of 112 bits, with an effort on the order of 2^{56}, which is not much more than the 2^{55} required for single DES.

TRIPLE DES WITH TWO KEYS

An obvious counter to the meet-in-the-middle attack is to use three stages of encryption with three different keys.This raises the cost of the meet-in-the-middle attack to 2^{112}, which is beyond what is practical now and far into the future. However,it has the drawback of requiring a key length of 56×3 = 168 bits,which may be somewhat unwieldy.

The function follows an encrypt-decrypt-encrypt (EDE) sequence

$$C = E(K_1, D(K_2, E(K_1, P)))$$

$$P = D(K_1, E(K_2, D(K_1, C)))$$

There is no cryptographic significance to the use of decryption for the second stage. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES:

$$C = E(K_1, D(K_1, E(K_1, P))) = E(K_1, P)$$

$$P = D(K_1, E(K_1, D(K_1, C))) = D(K_1, C)$$

3DES with two keys is a relatively popular alternative to DES and has been adopted for use in the key management standards ANS X9.17 and ISO 8732.

Currently, there are no practical cryptanalytic attacks on 3DES.Coppersmith [COPP94] notes that the cost of a brute-force key search on 3DES is on the order of $2^{112} \approx (5 \times 10^{35})$ and estimates that the cost of differential cryptanalysis suffers an exponential growth, compared to single DES, exceeding 10^{52}.

It is worth looking at several proposed attacks on 3DES that, although not practical, give a flavor for the types of attacks that have been considered and that could form the basis for more successful future attacks.

This method is an improvement over the chosen-plaintext approach but requires more effort. The attack is based on the observation that if we know A and C, then the problem reduces to that of an attack on double DES. Of course, the attacker does not know A, even if P and C are known, as long as the two keys are unknown. However, the attacker can choose a potential value of A and then try to find a known (P, C) pair that produces A. The attack proceeds as follows.

1. Obtain n(P, C) pairs.This is the known plaintext. Place these in a table sorted on the values of P.
2. Pick an arbitrary value for A, and create a second table with entries defined in the following fashion. For each 2^{56} of the possible keys $K_1 = i$, , calculate the plaintext value P_i that produces :

$$P_i = D(i, a)$$

For each P_i that matches an entry in Table 1,create an entry in Table 2 consisting of the K_1 value and the value of B that is produced for the (P, C) pair from Table 1, assuming that value of K_1 :

$$B = D(i, C)$$

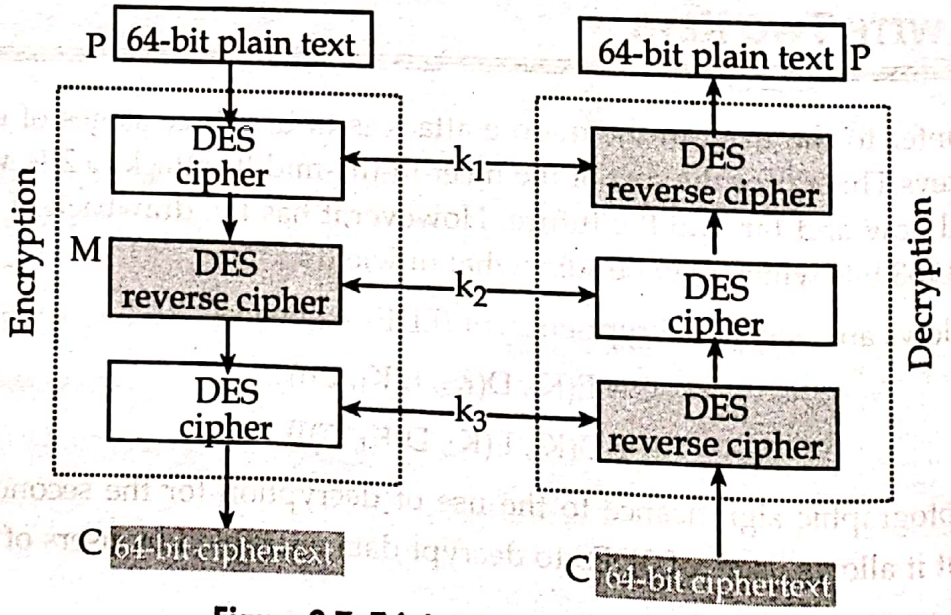


Figure 2.7: Triple DES with two Keys

3. We now have a number of candidate values of a position to search for a value of K_2 .For each of the 2^{56} possible keys $K_2 = j$, calculate the second intermediate value for our chosen value of a:

$$B_j = D(j, a)$$

4. Test each candidate pair of keys (i, j) on a few other plaintext–ciphertext pairs. If a pair of keys produces the desired ciphertext, the task is complete. If no pair succeeds, repeat from step 1 with a new value of a.

For a given known (P, C), the probability of selecting the unique value of a that leads to success is $1/2^{64}$. Thus, given (P, C) pairs, the probability of success for a single selected value of a is $n/2^{64}$. A basic result from probability theory is that the expected number of draws required to draw one red ball out of a bin containing n red balls and N – n green balls is $(N + 1)/(n + 1)$if the balls are not replaced. So the expected number of values of that must be tried is, for large n,

$$\frac{2^{64} + 1}{n+1} \approx \frac{2^{64}}{n}$$

Thus the expected running time of the attack is on the order of

$$(2^{56})\frac{2^{64}}{n} = 2^{120-\log n}$$

TRIPLE DES WITH THREE KEYS

Although the attacks just described appear impractical, anyone using two-key 3DES may feel some concern. Thus, many researchers now feel that three-key 3DES is the preferred alternative (e.g.,[KALI96a]).Three-key 3DES has an effective key length of 168 bits and is defined as

$$C = E(K_3, D(K_2, E(K_1, P)))$$

Backward compatibility with DES is provided by putting $K3 = K_2$ or $K_1 = K_2$.

FINITE FIELDS

GROUPS

A **group** G, sometimes denoted by {G, ·} is a set of elements with a binary operation, denoted by, that associates to each ordered pair (a, b) of elements in G an element (a · b) in G, such that the following axioms are obeyed. The operator · is generic and can refer to addition, multiplication, or some other mathematical operation.

(A1) Closure: If a and b belong to G, then a · b is also in G.

(A2) Associative: a · (b · c) = (a · b) · c for all a, b, c in G.

(A3) Identity element: There is an element e in G such that a · e = e · 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 · a' = a' · 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.

CYCLIC GROUP

We define exponentiation within a group as repeated application of the group operator, so that $a^3 = a \cdot a$

a. Further, we define $a^0 = e$, the identity element; and $a^{-n} = (a')^n$. 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.

RING

A **ring** R, sometimes denoted by {R, +, x}, is a set of elements with two binary operations, called addition and multiplication, 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.

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

Commutative 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.

INTEGRAL DOMAIN

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, +, x}, 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:

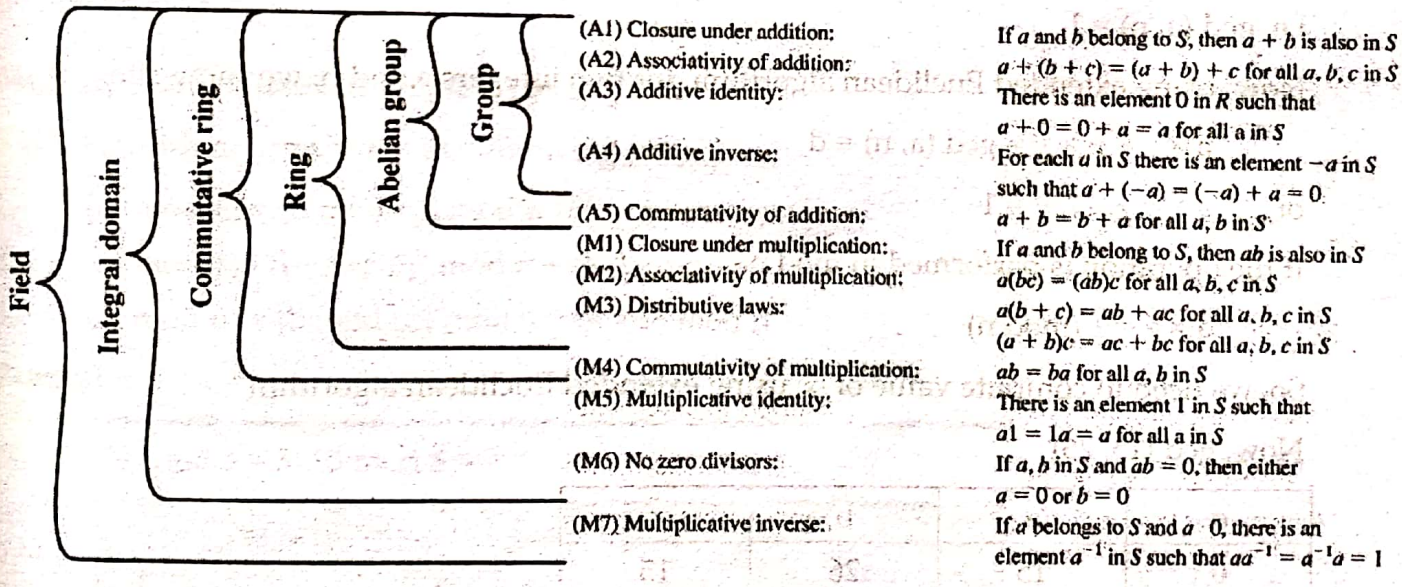
(A1M6) F is an integral domain; that is, F satisfies axioms A1 through A5 and M1 through M6.

(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.



MODULAR ARITHMETIC

Given any positive integer n and any nonnegative integer a, if **we divide a by n**, we get an integer quotient **q** and an integer remainder **r** that obey the following relationship:

$$a = qn + r \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

If a is an integer and n is a positive integer, we define **a mod n** to be the remainder when a is divided by n. The integer n is called the **modulus**.

For example: 11 mod 7 = 4 and -11 mod 7 = 3

Two integers a and b are said to be **congruent modulo n,** if (a mod n) = (b mod n).

If "a" is an integer and "n" is a positive number we defined a mod n to be the remainder when "a" is divided by "n" so we say $a = qn + r, 0 \leq r < n; q = (\lfloor a/n \rfloor \times n) + (a \bmod n)$

- The operation has two input a and n and two output q and r:

Set of residue Zn

- Define Zn as a set of non-negative integers less then n Zn = {0, 1, 2, ..., n-1}
- This is required to as a set of residues or residue class m an integers between 0 to n – 1.
- The result of a (mod n) is always a non-negative integers less than n.
- So modulo operation creates a set which in modular arithmetic referred to as set of least residues modulo n or Zn.
- Using extended euclidean algorithm to find multiplicative inverse.

Example: *Find multiplicative inverse of 15 in* Z_{26}.

Solution:

Any positive integers less than 'n' and relatively prime to 'n' has a multiplicative inverse mod n.

To compute multiplicative inverse a (mod n), the gcd of a and n should be 1.

i.e. gcd (a, n) = 1

Now, using extended Euclidean algorithm, for two integers a and n, we can write,

$a \times x + n \times y = \gcd(a, n) = d$

or, $a \times x + n \times y = 1$

If the operation is performed in mod n,

$a \times x = 1 \pmod n$

So, we need to compute value of 'x' using extended Euclidean algorithm

Now, gcd (15, 26)

q	a	b	r
0	15	26	15
1	26	15	11
1	15	11	4
2	11	4	3
1	4	3	1
03	3	1	0

∴ gcd (15, 26) = 1

DIVISORS

We say that a nonzero b divides a if a = mb for some m, where a, b, and m are integers. That is, b divides a if there is no remainder on division. The notation is commonly used to mean b divides a. Also, if b | a, we say that b is a divisor of a.

The following relations hold:

- If a | 1, then a = ±1.
- If a | b and b | a, then a = ±b.
- Any b 0 divides 0.
- If b | g and b | h, then b | (mg + nh) for arbitrary integers m and n.

Properties of Congruences

Congruences have the following properties:

1. a b (mod n) if n | (a-b).
2. a b (mod n) implies b a (mod n)..
3. a b (mod n) and b c (mod n) imply a c (mod 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 mod n) = (reminder when b + kn is divided by n) = (reminder when b is divided by n) = (b mod n)

MODULAR ARITHMETIC OPERATIONS

Modular arithmetic exhibits the following properties:

1. [(a mod n) + (b mod n)] mod n = (a + b) mod n
2. [(a mod n) - (b mod n)] mod n = (a- b) mod n
3. [(a mod n) x (b mod n)] mod n = (a x b) mod n

Examples

11 mod 8 = 3; 15 mod 8 ≡ 7

[(11 mod 8) + (15 mod 8)] mod 8 = 10 mod 8 =
2 (11 + 15) mod 8 = 26 mod 8 = 2

[(11 mod 8) (15 mod 8)] mod 8 = 4 mod 8 = 4

(11 15) mod 8 = 4 mod 8 = 4

[(11 mod 8) x (15 mod 8)] mod 8 = 21 mod 8
= 5 (11 x 15) mod 8 = 165 mod 8 = 5

Exponentiation is performed by repeated multiplication, as in ordinary arithmetic.

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

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

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

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

ARITHMETIC MODULO 8

+	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	2	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) Multiplicative 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 inverse modulo 8

Here –w is additive inverse of w and w^{-1} is the multiplicative inverse of w.

Define the set Z_n as the set of nonnegative integers less than n $Z_n = \{0, 1, 2, 3, \ldots\ldots n-1\}$.

This is referred to as the *set of residues*, or residue classes modulo n. To be more precise, each integer in Z_n represents a residue class. We can label the residue classes modulo n as *[0]*, [1], [2],...,[n 1], where

[r] = {a: a is an integer, a r (mod n)}

The residue classes *modulo* 4 are

[0] = { ..., 16, 12, 8, 4, 0, 4, 8, 12, 16,... }

[1] = { ..., 15, 11, 7, 3, 1, 5, 9, 13, 17,... }

[2] = { ..., 14, 10, 6, 2, 2, 6, 10, 14, 18,... }

[3] = { ..., 13, 9, 5, 1, 3, 7, 11, 15, 19,... }

If we perform modular arithmetic within Z_n, the properties shown in Table (below) hold for integers in Z_n. Thus, *Z_n is a commutative ring with a multiplicative identity element.*

Commutative laws	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
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 laws	$[w + (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$ $[w + (x \times y)] \bmod n = [(w + x) \times (w + y)] \bmod n$
Identities	$(0 + w) \bmod n = w \bmod n$ $(1 + w) \bmod n = w \bmod n$
Additive inverse (-w)	For each $w \in Zn$, there exists a z such that $w + z \equiv 0 \bmod n$

EXTENDED EUCLIDEAN ALGORITHM

Let a and b are two positive integers. Then gcd (a, b) can be expressed as linear combination with integer coefficients of a and b i.e. if a and b are positive integer then there exist integer s and f such that

$$\gcd(a, b) = sa + tb$$

Example: *gcd (2740, 1760)*

1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

∴ gcd (2740, 1760) = 20

Example: *Express gcd (252, 198) = 18 as a linear combination of 252 and 198.*

Solution:

Using Enclidean algorithm,

252 = 1.198 + 54

198 = 3.54 + 36

54 = 1.36 + 18

36 = 2 * 18

Using the next to last division, we can express gcd (252, 198) = 18 as libeal combinations i.e.

18 = 54 - 1 * 36 .....(i)

where, s = 1 and t = - 1

Similarly, from second division,

36 = 198 - 3 * 54 .....(ii)

From equation (i) and (ii) [Substitute value of 36 into (i)]

18 = 54 - 1(198 - 3 * 54)

= 4 * 54 - 198 .....(iii)

From first division, we have 54 = 252 - 1 * 198 .....(iv)

Now, substitute value of 54 from (iv) to (iii), we have

18 = 4 * [252 - 198] - 198

= 4 * 252 - 5 * 198 which is of the form

18 = 5 * 252 + t * 198 with

s = 4 & t = - 5

Example: *Express the gcd of 33 & 44 as linear combination of these integer.*

Solution:

Using Endidean Algorithm

$44 = 1 * 33 + 11$

$33 = 3 * 21$

From next to last division,

$11 = 44 - 1 * 33$

which is in the form of

$11 = 5 * 44 + t * 33$

with $s = 1$ & $t = -1$

Find 54t when gcd (124, 232) = 1

Using Endedian algo

$232 = 1.124 + 108$ .....(i)

$124 = 1 * 108 + 16$ .....(ii)

$108 = 6 * 16 + 12$ .....(iii)

$16 = 1 * 12 + 4$ .....(iv)

$12 = 3 * 4$ .....(iv)

From next to last division,

$4 = 16 - 1 * 12$ .....(v)

From third step above,

$12 = 108 - 6 * 16$ .....(vi)

From (v) and (vi), we have

$4 = 16 - [108 - 6 * 16]$

$4 = 16 - 108 + 6 * 16$

$= 7 * 16 - 108$ .....(vi)

From 2nd steps,

$16 = 124 - 1 * 108$ .....(viii)

Again, substituting value from (viii) to (vi)

$4 = 7[124 - 1 * 108] - 108$

$= 7 * 124 - 8 * 108$ .....(ix)

Finally, from first step,

$108 = 232 - 1 * 124$ .....(x)

Now, from (ix) and (x),

$4 = 7 * 124 - 8[232 - 1 * 124]$

$4 = 7 * 124 - 8 * 232 + 8 * 124$

$= 15 * 124 - 8 * 232$

which is in the form of

$4 = 5 * 124 + t * 232$

where,

$s = 15$ & $t = -8$

FINITE FIELDS OF ORDER P

For a given prime, p, the finite field of order p, GF(p) is defined as the set Z_p of integers {0, 1,..., p- 1}, together with the arithmetic operations modulo p. (GF stands for Galois field)

Arithmetic in GF(7)

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

(a) Addition modulo 7

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

(b) Multiplicative modulo 8

w	-w	w^{-1}
0	0	.....
1	6	1
2	5	4
3	4	5
4	3	2
5	2	3
6	1	6

(c) Additive and multiplicative inverse modulo 7

Finding the Multiplicative Inverse in GF(p)

It is easy to find the multiplicative inverse of an element in GF(p) for small values of p. You simply construct a multiplication table, such as shown in Table above, and the desired result can be read directly. However, for large values of p, this approach is not practical.

If gcd(m, b) = 1, then b has a multiplicative inverse modulo m. That is, for positive integer b < m, there exists a b1 < m such that bb1 = 1 mod m. The Euclidean algorithm can be extended so that, in addition to finding gcd(m, b), if the gcd is 1, the algorithm returns the multiplicative inverse of b.

Extended Euclid (M, B)

1. (A1, A2, A3) ← (1, 0, m); (B1, B2, B3) ← (0, 1, b)
2. if B3 = 0 return A3 = gcd(m, b); no inverse
3. if B3 = 1 return B3 = gcd(m, b); B2 = b_1 mod m
4. Q = ⌊A3/B3⌋
5. (T1, T2, T3) ← (A1 QB1, A2 QB2, A3 QB3)
6. (A1, A2, A3) ← (B1, B2, B3)
7. (B1, B2, B3) ← (T1, T2, T3)
8. goto 2

Exercise: trace the above algorithms for finding multiplicative inverse of 550 in GF(1759)

POLYNOMIAL ARITHMETIC

Before continuing our discussion of finite fields, we need to introduce the interesting subject of polynomial arithmetic. We are concerned with polynomials in a single variable, and we can distinguish three classes of polynomial arithmetic.

- Ordinary polynomial arithmetic, using the basic rules of algebra.
- Polynomial arithmetic in which the arithmetic on the coefficients is performed modulo p; that is, the coefficients are in FG(p).
- Polynomial arithmetic in which the coefficients are in GF(p), and the polynomials are defined modulo a polynomial whose highest power is some integer n.

ORDINARY POLYNOMIAL ARITHMETIC

A polynomial of degree (integer $n \geq 0$) is an expression of the form

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

where the a_i are elements of some designated set of numbers S, called the coefficient set, and $a_n \neq 0$. We say that such polynomials are defined over the coefficient set S.

A zero-degree polynomial is called a constant polynomial and is simply an element of the set of coefficients. An nth-degree polynomial is said to be a monic polynomial if $a_n = 1$.

In the context of abstract algebra, we are usually not interested in evaluating a polynomial for a particular value of x. To emphasize this point, the variable x is sometimes referred to as the indeterminate.

Polynomial arithmetic includes the operations of addition, subtraction, and multiplication. These operations are defined in a natural way as though the variable x was an element of S. Division is similarly defined, but requires that S be a field. Examples of fields include the real numbers, rational numbers, and Z_p for p prime. Note that the set of all integers is not a field and does not support polynomial division.

Addition and subtraction are performed by adding or subtracting corresponding coefficients. Thus, if

$$f(x) = \sum_{i=0}^{n} a_i x^i,\ g(x) = \sum_{i=0}^{m} b_i x^i;\ n \geq m$$

Then addition is defined as

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i)x^i + \sum_{i=m+1}^{n} b_i x^i$$

And multiplication is defined as

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

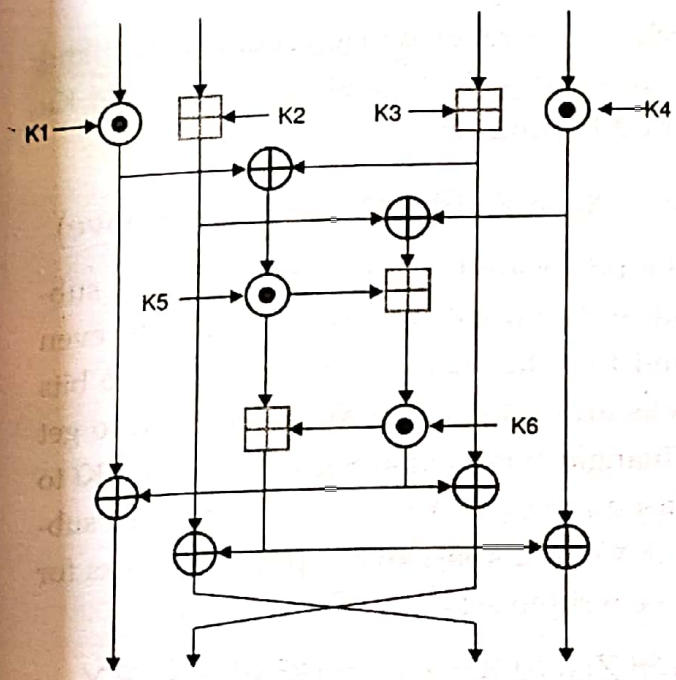
Where, $c_k = a_0 b_k = a_1 b_{k-1} + \ldots + a_{k-1} b_1 + a_k b_0$

In the last formula, we treat a as zero for $i > n$ and b_i as zero for $I > m$ .Note that the degree of the product is equal to the sum of the degrees of the two polynomials.

INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA)

In cryptography, the IDEA is a block cipher designed by Xuejia Lai and James Massey and was first described in 1991. The algorithm was intended as a replacement for the Data Encryption Standard.

Operation: IDEA operates on 64-bit blocks using a 128-bit key, and consists of a series of eight identical rounds (see figure below) and an output round (the half-round, see figure below). The processes for encryption and decryption are similar. IDEA derives much of its security by interleaving operations from different groups - modular addition (addition mod 2^{16}, denoted by ⊞), modular multiplication (multiplication mod 2^{16} +1 denoted by ⊙, where the all-zero word (0 x 0000) is interpreted as 2^{16}),, and bitwise eXclusive OR (XOR) (denoted by ⊕), - which are algebraically "incompatible" in some sense. The blow figures in left shows a round (1-8) and in right shows final "half round.



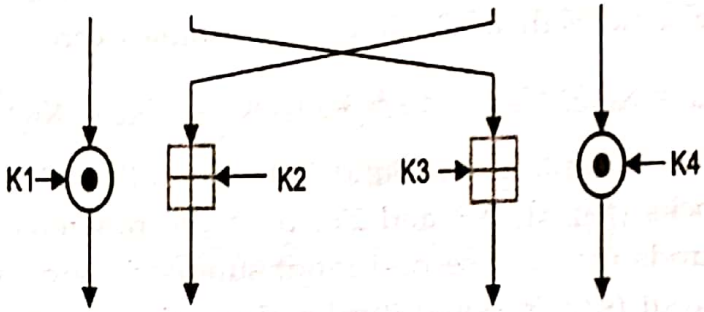


Figure 2.8: An Encryption Round of IDEA

Key schedule: The 128 bit key is used to generate 52 sub keys of length 16 bits. 6 keys are used for each round and the remaining 4 keys are used in final half round. The figure below shows the key generation mechanisms where we start getting the subkeys from the starting position of the 128 bits key, so in first round we get 8 subkeys. In each of the round of sub key generation 128 bits keys undergoes a 25 bit position right to generate next set of keys i.e. start with bit position 0, 25, 50 and so on until all 52 keys are derived.

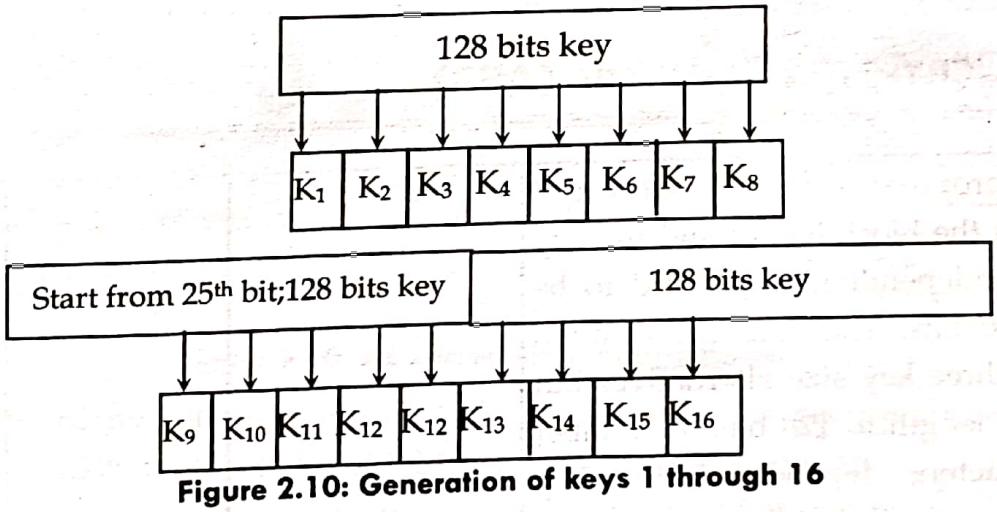


Figure 2.10: Generation of keys 1 through 16

Round Operations: It has been mentioned above that IDEA uses 8 full rounds and 1 half round. We now break the 8 full round and make it 16 rounds such that there are total 17 rounds where 9 odd rounds (1, 3, …., 17) are identical and 8 even rounds (2, 4, ….,16) are identical. Each odd round takes 4 subkeys and each even round takes 2 subkeys.

Odd Round: Odd round is simple and uses four keys where first and fourth keys are uses multiplication mod $2^{16} + 1$ operation with first and fourth 16 bits fragment of 64 bits input block respectively. The second and third subkeys are subjected to addition mod 2^{16} with second and third bits fragment of 64 bits input block respectively. The output so obtained from operations with first and fourth input sub-block will be first and fourth input for next round and output

from operations with second and third input sub-block will be reversed i.e. becomes third and second input for next round. If X_a, X_b, X_c, and X_d are input sub-blocks and K_a, K_b, K_c, and K_d are subkeys then we can write algebraic expression for odd round as:

$X_a = X_a \odot K_a$; $X_b = X_c \boxplus K_c$; $X_c = X_b \boxplus K_b$; $X_d = X_d \odot K_d$; (see figure above)

Even Round: Even round is bit complicated then the odd round. There are four input sub-blocks (X_a, X_b, X_c, and X_d) from the previous round and two subkeys (K_e and K_f). In even rounds first and second input sub-blocks are subjected to XOR operation to get single 16 bits output (say, Y_{in}) and third and fourth input sub-blocks are subjected to XOR operation to get single 16 bits output (say, Z_{in}). Y_{in} and Z_{in} are fed to mangler function along with K_e, and K_f to get outputs Y_{out} and Z_{out}, where Y_{out} is XOR'ed with X_a and X_b, to get first two input sub-blocks for next round and Z_{out} is XOR'ed with X_c and X_d, to get last two input sub-blocks for next round. Algebraic expressions for even round can be written as:

$Y_{in} = X_a \oplus X_b$; $Z_{in} = X_c \oplus X_d$; $Y_{out} = ((K_e \odot Y_{in}) \boxplus Z_{in}) \odot K_f$; $Z_{out} = (K_e \odot Y_{in}) \boxplus Y_{out}$;
$X_a = X_a \oplus Y_{out}$; $X_b = X_b \oplus Y_{out}$; $X_c = X_b \oplus Z_{out}$; $X_d = X_d \oplus Z_{out}$; (see figure above)

Security: The designers analyzed IDEA to measure its strength against differential cryptanalysis and concluded that it is immune under certain assumptions. No successful linear or algebraic weaknesses have been reported. Some classes of weak keys have been found but these are of little concern in practice, being so rare as to be unnecessary to avoid explicitly. As of 2004, the best attack which applies to all keys can break IDEA reduced to 5 rounds (the full IDEA cipher uses 8.5 rounds).

ADVANCED ENCRYPTION STANDARD (AES)

The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. A number of AES parameters depend on the key lengthKey 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	12	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

AES ENCRYPTION DECRYPTION TECHNIQUE

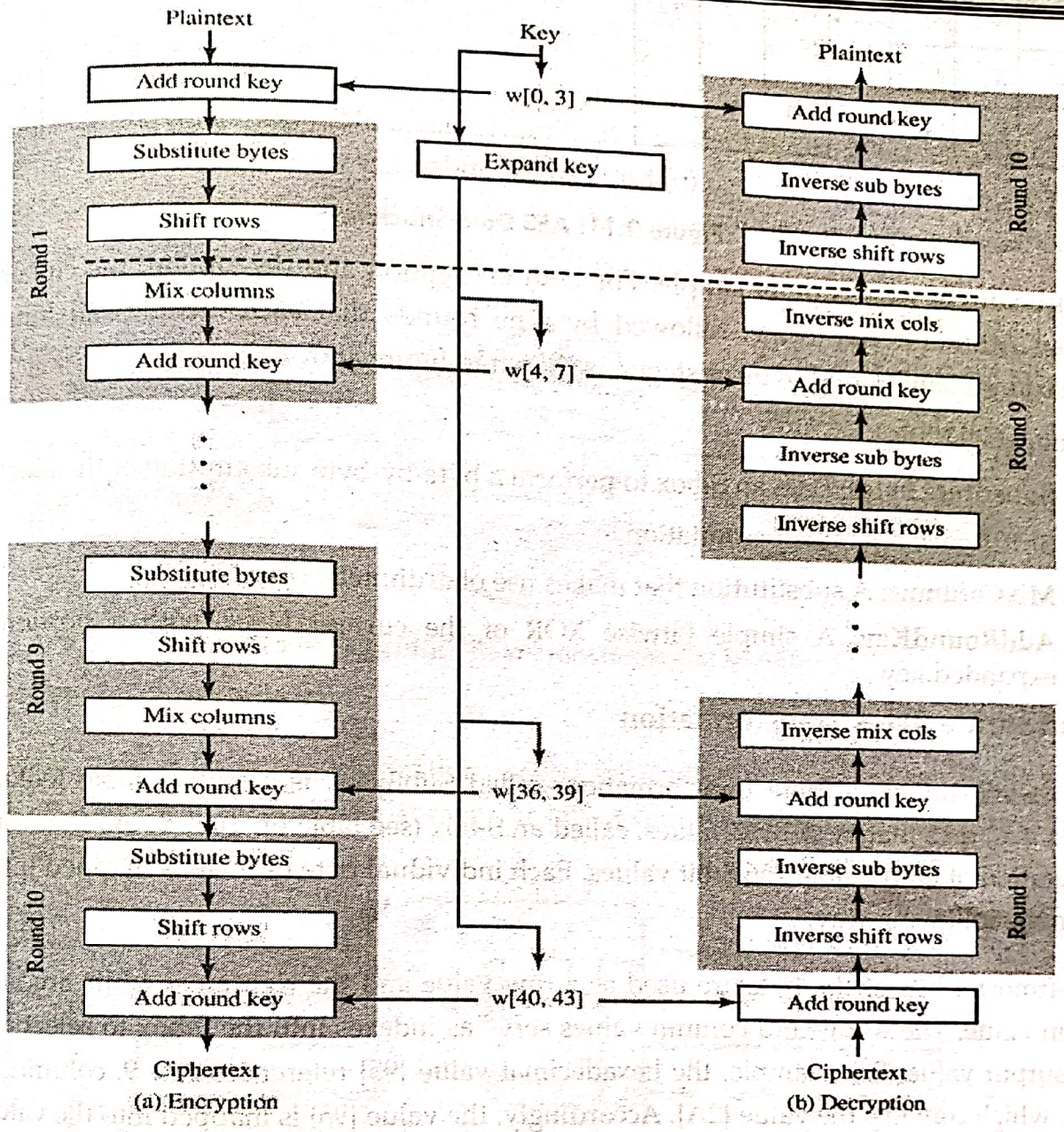
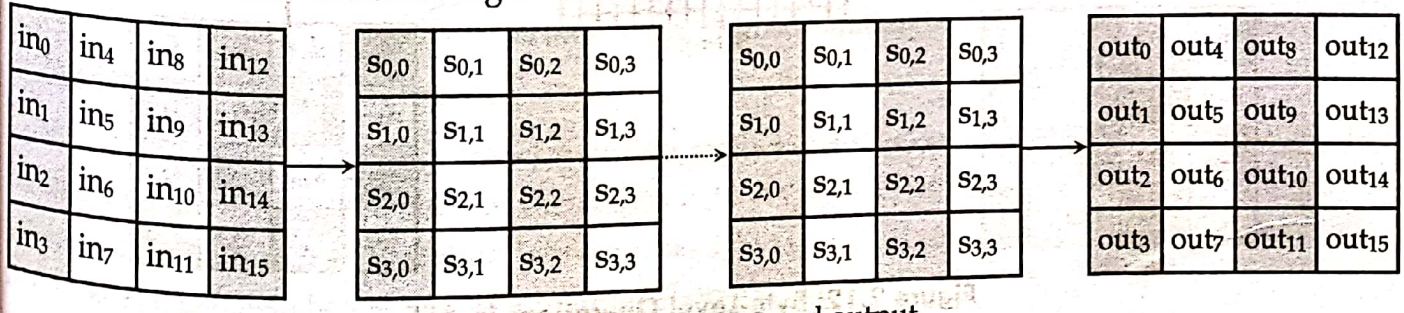


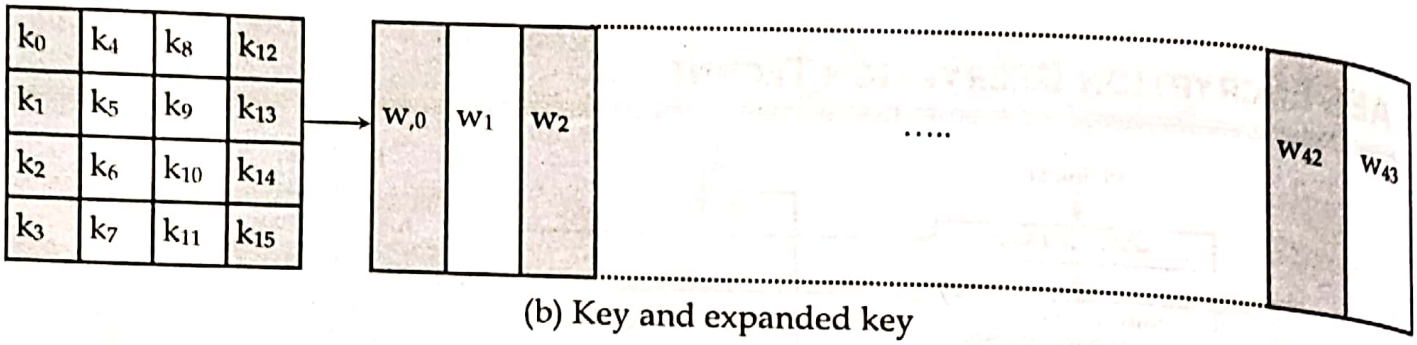
Figure 2.10: AS Encryption and Decryption

This figure shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix.

AES data structures: the following matrixes are used in AES encryption and Decryptions



(a) Input, state array, and output



(b) Key and expanded key

Figure 2.11: AES Data Structures

The structure of AES 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 as shown in figure above.

The four stages are:

- **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 GF(2^8)
- **AddRoundKey:** A simple bitwise XOR of the current block with a portion of the expanded key

a) Substitute Bytes Transformation

The forward substitute byte transformation, called SubBytes, is a simple table lookup. AES defines a 16 x 16 matrix of byte values, called an S-box (see table on Text Book) 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 of the S-box, which contains the value {2A}. Accordingly, the value {95} is mapped into the value {2A}.

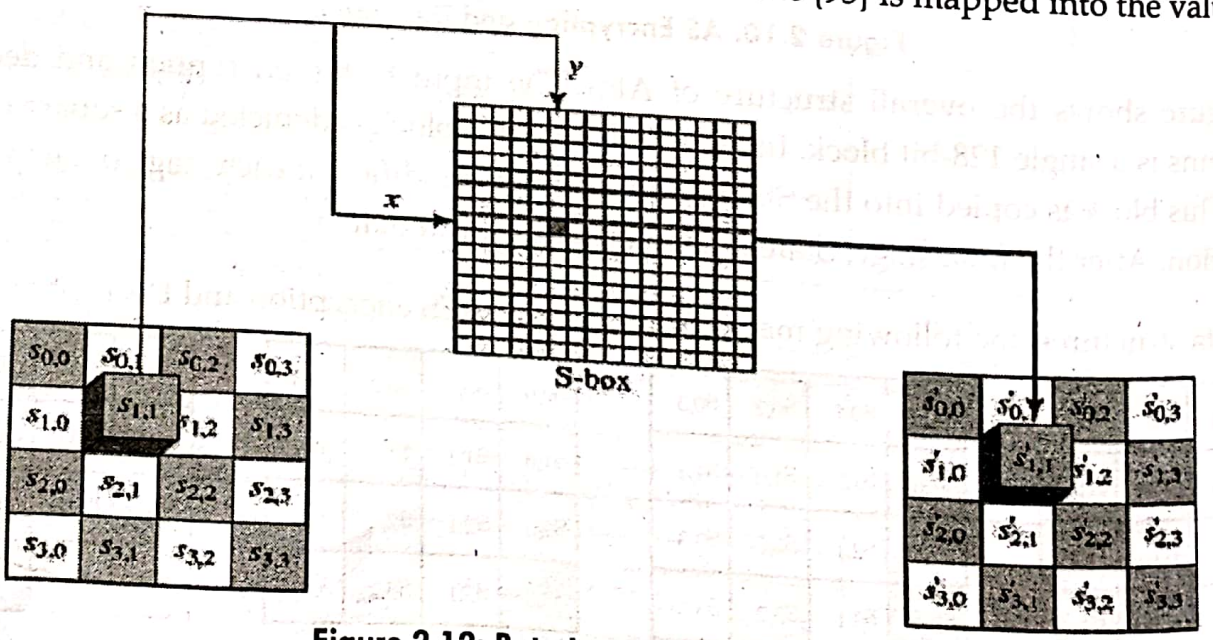


Figure 2.12: Byte-Level Operations in AES

The inverse substitute byte transformation, called InvSubBytes, makes use of the inverse S-box. For example, that the input {2A} produces the output {95} and the input {95} to the S-box produces {2A}.

b) Shift Rows Transformation

The forward shift row transformation, called ShiftRows, is depicted in Figure below. 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 following is an example of ShiftRows:

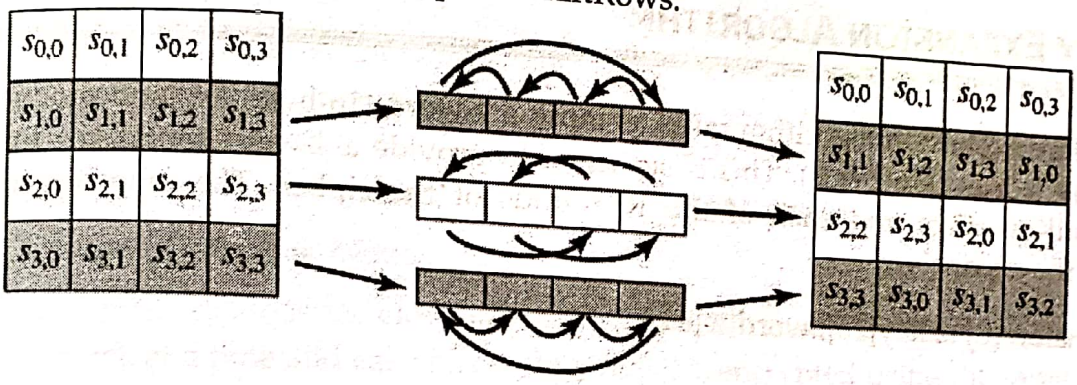


Figure 2.12: Shift Row Transformation in AES

c) Mix Columns Transformation

The forward mix column transformation, called MixColumns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column.

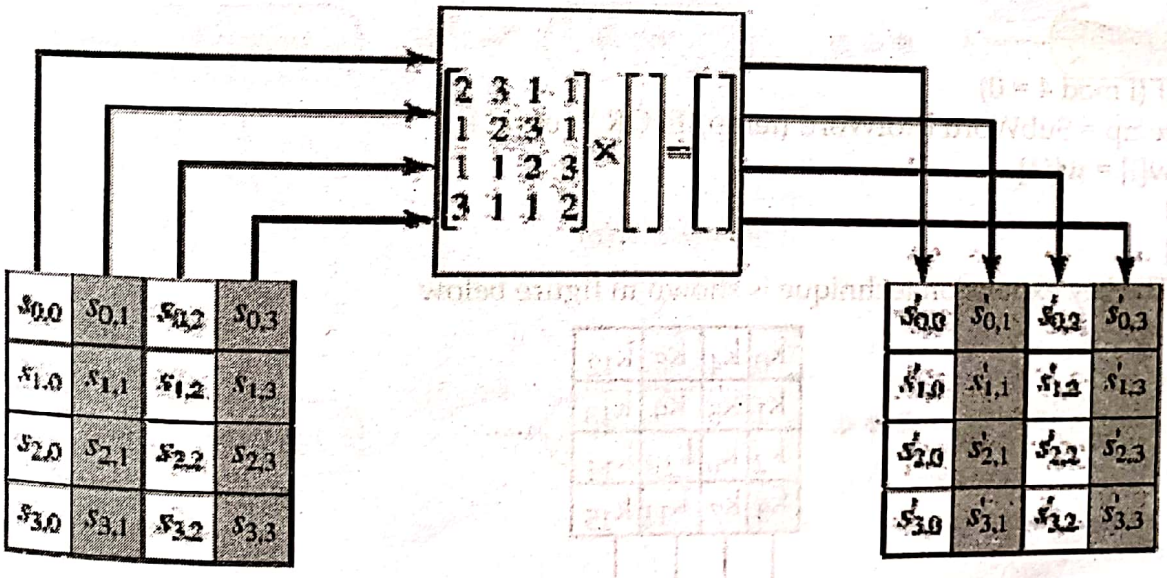


Figure 2.13: Mix column transformation in AES

d) Add RoundKey Transformation

In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key. As shown in Figure below, 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.

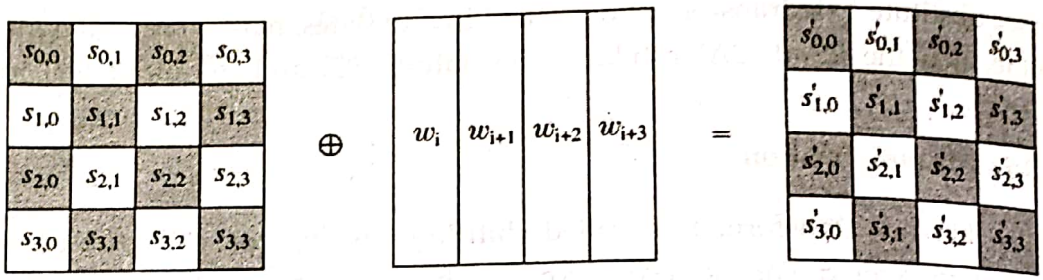


Figure 2.14: Add Roundkey transformation in AES

AES KEY EXPANSION ALGORITHM

The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The following pseudocode describes the expansion:

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

        {
        temp = w[i 1];


        if (i mod 4 = 0)
        temp = SubWord (RotWord (temp))ExOR Rcon[i/4];
        w[i] = w[i4]
        }
        }
```

The key expansion technique is shown in figure below

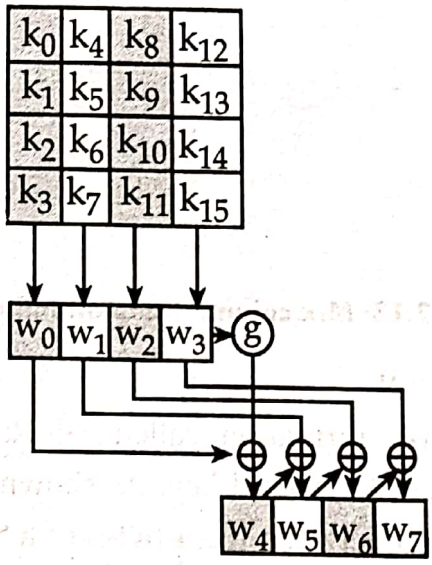


Figure 2.15: AES Key Expansion

Where g is a special function consist of following sub functions

1. RotWord performs a one-byte circular left shift on a word. This means that an input word $[b_0, b_1, b_2, b_3]$ is transformed into $[b_1, b_2, b_3, b_0]$.
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].

The round constant is a word in which the three rightmost bytes are always 0.(See the text book for more details)

MODES OF OPERATIONS

A mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. The four modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used.

1. Electronic Codebook Mode

The simplest mode is the electronic codebook (ECB) mode, 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 b-bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showing its corresponding ciphertext.

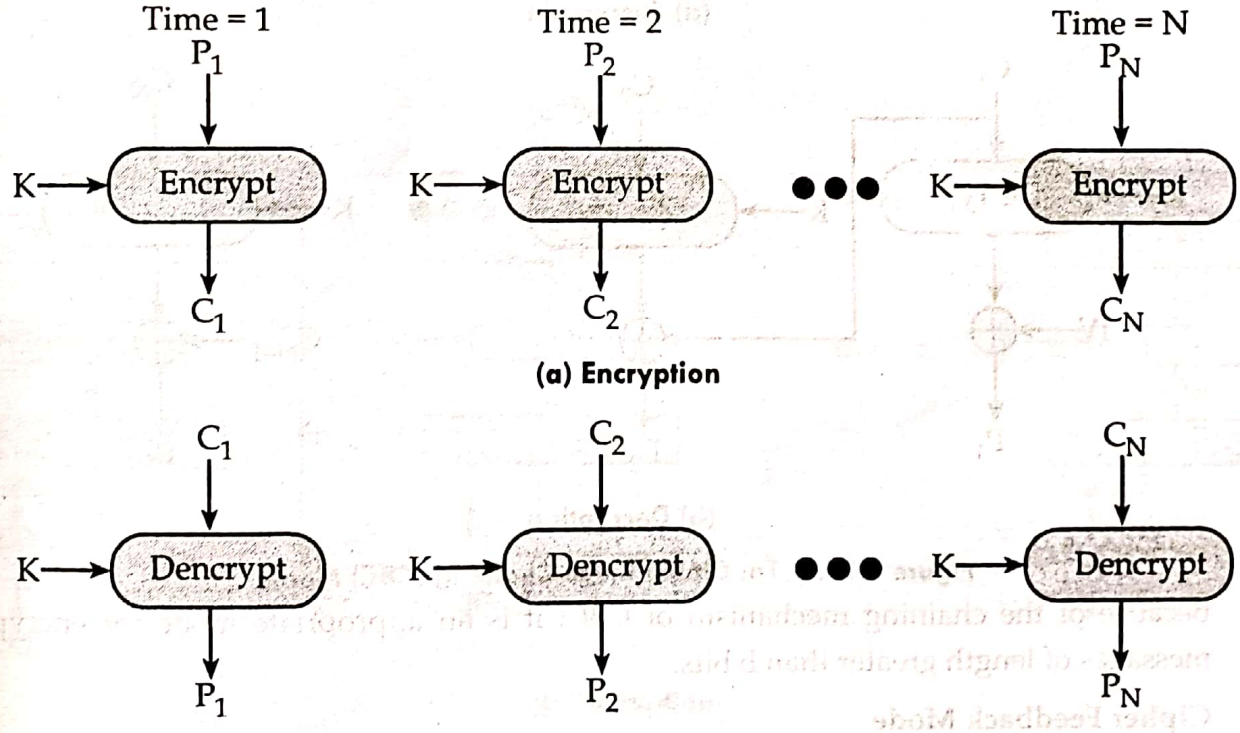


Figure 2.16: Electronic Codebook (ECB) Mode

The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES key securely, ECB is the appropriate mode to use. The most significant characteristic of ECB is that the same b-bit block of plaintext, if it appears more than once in the message, always produces the same ciphertext. For lengthy messages,

the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities.

2. **Cipher Block Chaining Mode**

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode. In this scheme, 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.

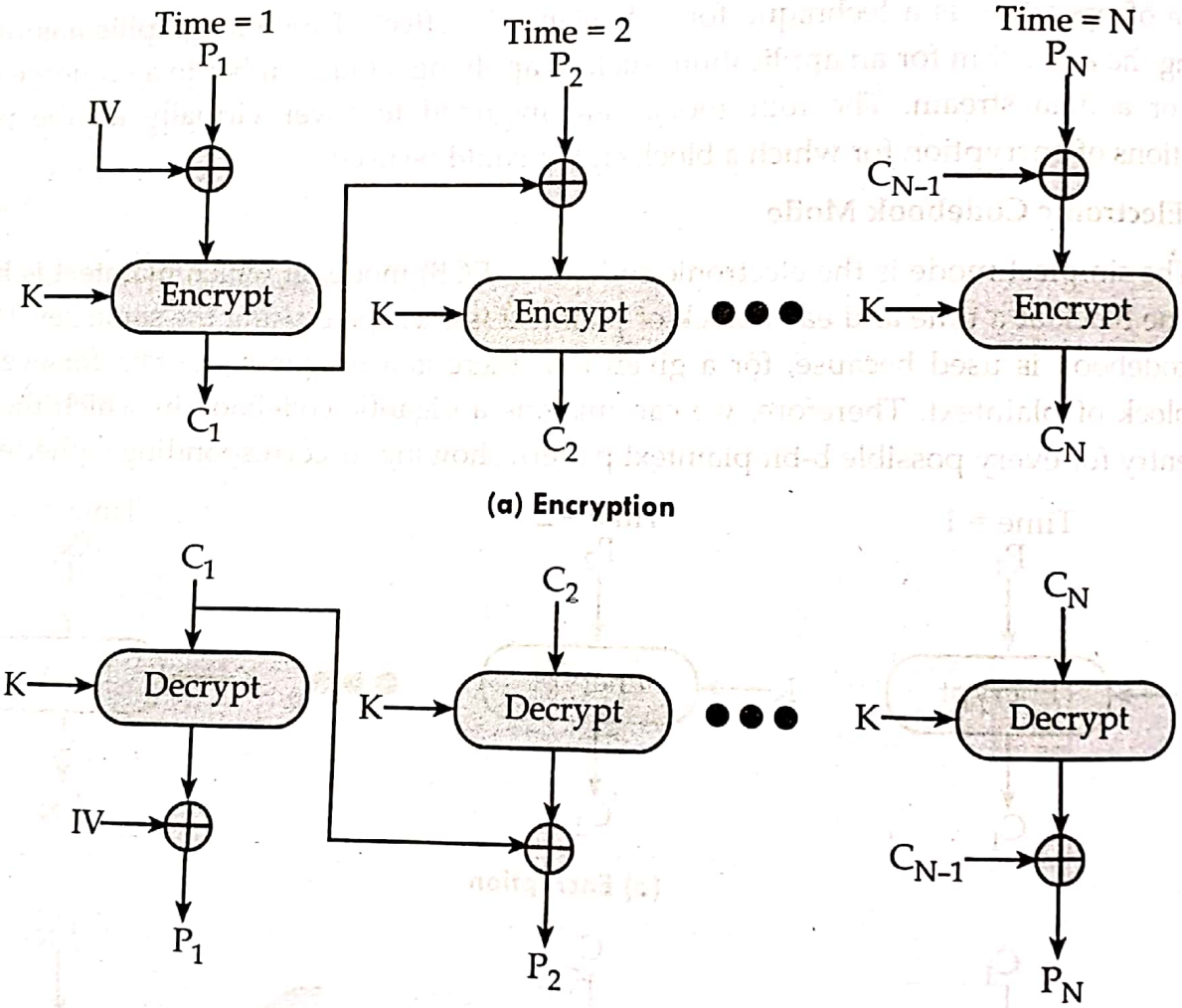


Figure 2.17: 2.16: Cipher Block Chaining (CBC) Mode

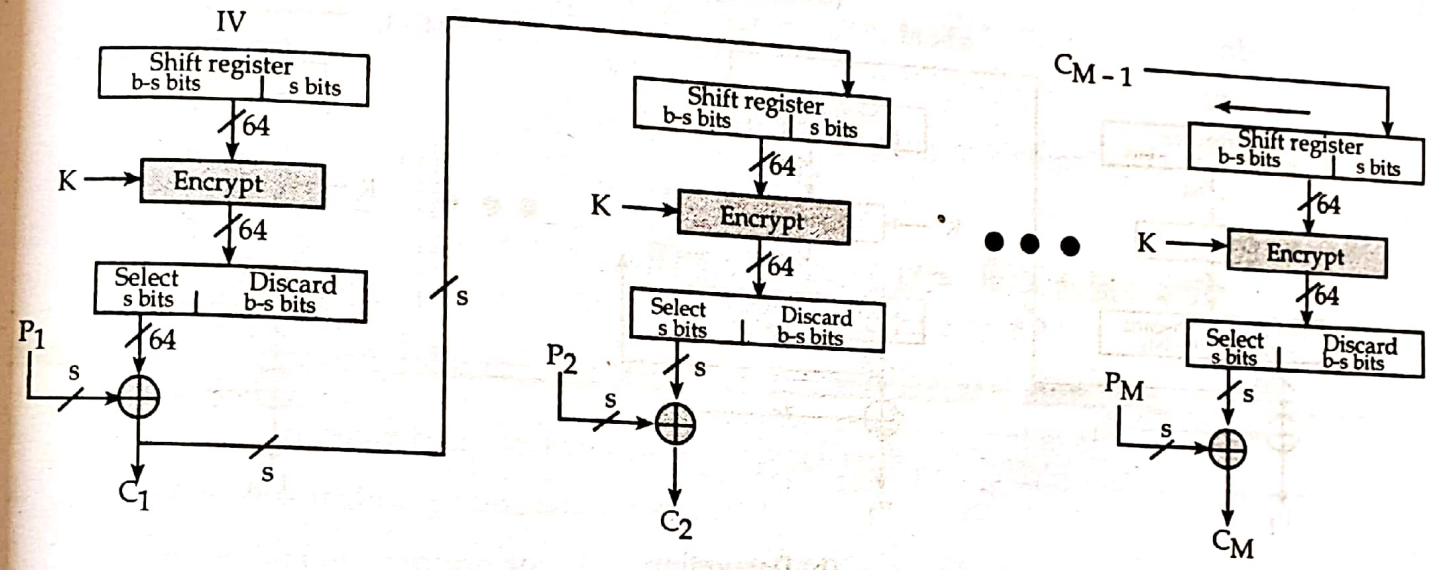
Because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than b bits.

3. **Cipher Feedback Mode**

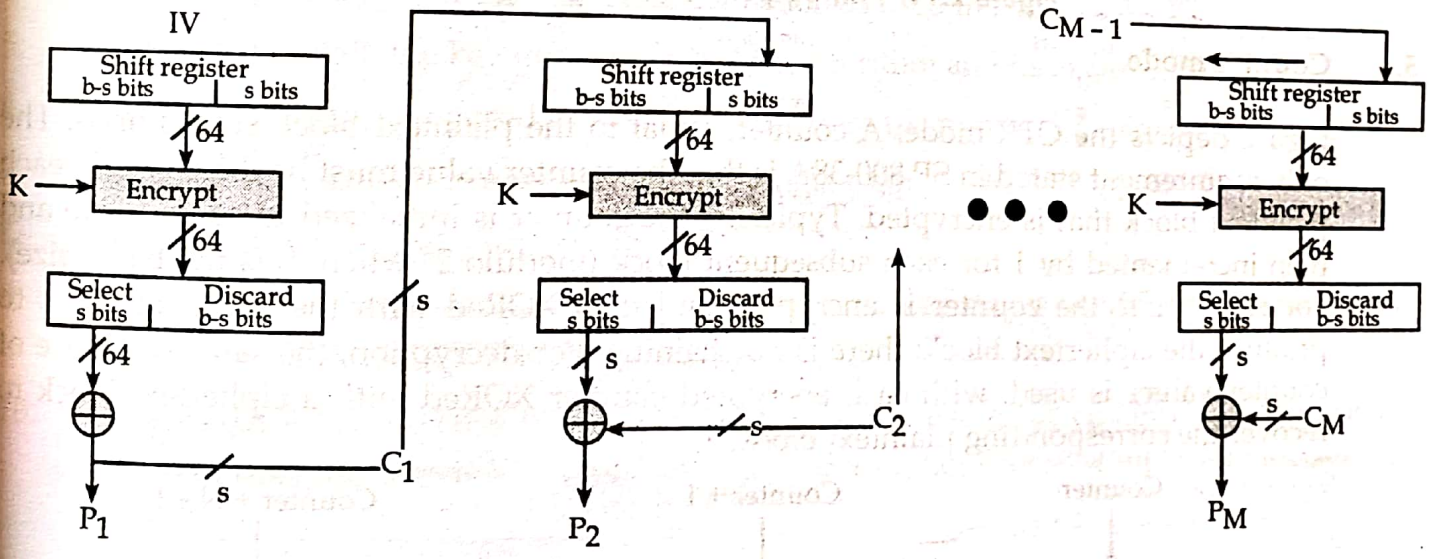
The DES scheme is essentially a block cipher technique that uses b-bit blocks. However, it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a cipher text output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.

In the figure, it is assumed that 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 units of b bits, the plaintext is divided into segments of s bits.



(a) Encryption

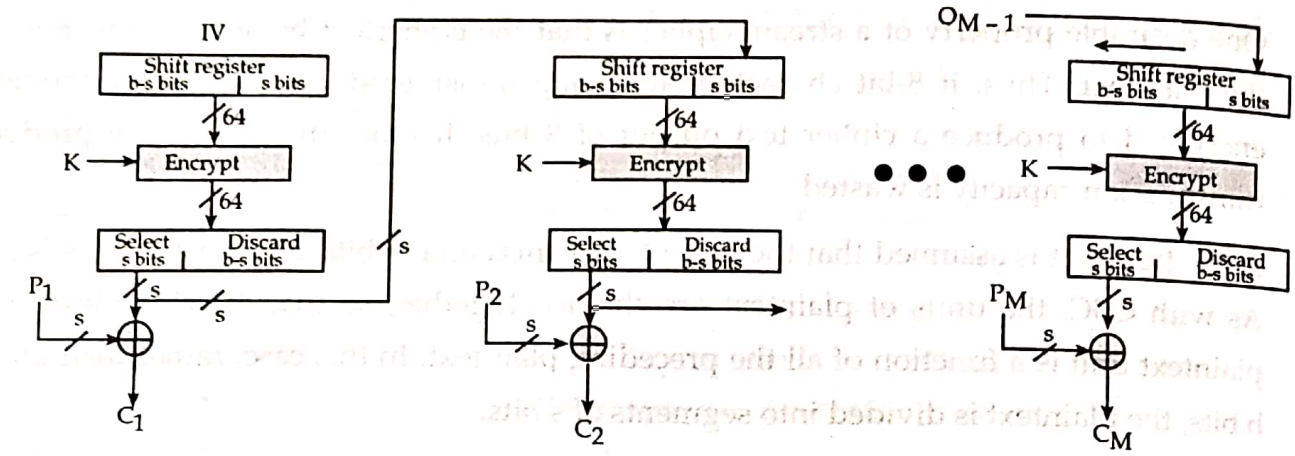


(b) Decryption

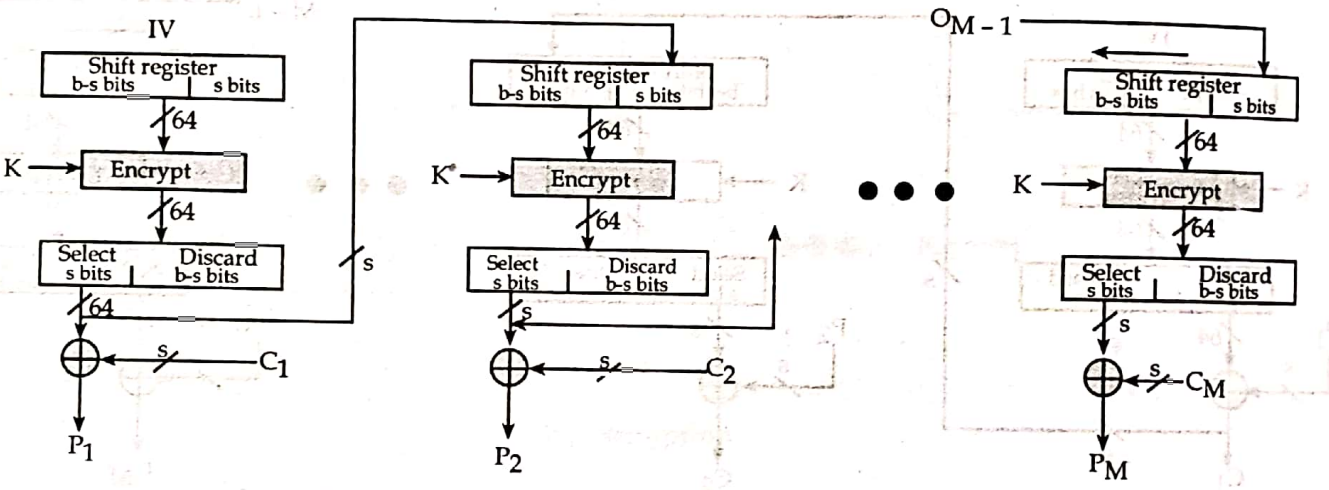
Figure 2.18: s-bit Cipher Feedback (CFB) Mode

4. Output Feedback Mode

The output feedback (OFB) mode is similar in structure to that of CFB, as illustrated in figure. As can be seen, it is 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.



(a) Encryption

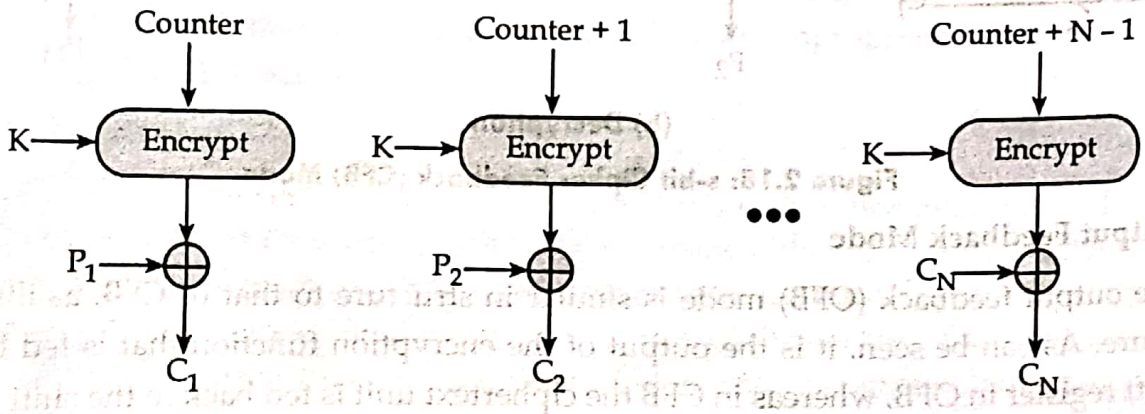


(b) Decryption

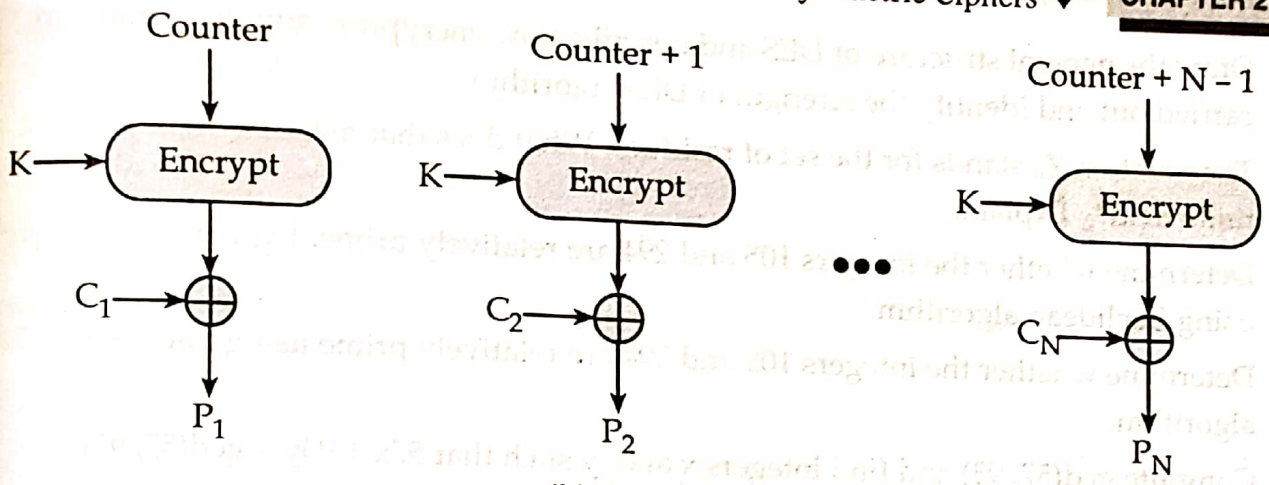
Figure 2.19: s-bit Output Feedback (OFB) Mode

5. Counter mode

Figure depicts the CTR mode. A counter, equal to the plaintext block size is used. The only requirement stated in SP 800-38A is that the counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block (modulo 2^b where b 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.



(a) Encryption



(b) Decryption

Figure 2.20: Counter (CTR) Mode

DISCUSSION EXERCISE

1. What do you mean by feistel structure for block ciphers? Explain
2. What is substitution permutation network?
3. Describe data encryption standard. Describe F- function in DES.
4. Explain the concepts of diffusion and confusion as used in DES.
5. Describe in detail, the key generation in AES algorithm and its expansion format.
6. How many rounds are used in AES and what does the number of rounds depend on?
7. What are four steps that are executed in a single round of AES Processing?
8. Describe subbytes and shiftrows in AES.
9. Briefly describe about MixColumns and AddRound key stages in AES. How many bytes in a state are affected by shiftRows round?
10. What is IDEA? How IDEA operates on 64- bits blocks using 128 bit keys? Describe each round of operations that IDEA follows to generate cipher text of 64 bit input message block.
11. Describe Triple DES and its applications.
12. What do you mean by S-Box? Explain the purpose of S-Box in DES.
13. Differentiate between double DES and Triple DES
14. What are different modes of block cipher encryption? Explain each of them in brief.
15. Formulate the single round of DES algorithm and design the key discarding process of DES.

16. Draw the general structure of DES and describe how encryption and decryption are carried out and identify the strength of DES algorithm.
17. The notation Z_n stands for the set of residues. What does that mean? Why is Z_n not a finite field? Explain.
18. Determine whether the integers 105 and 294 are relatively prime. Explain your answer using Euclidean algorithm.
19. Determine whether the integers 105 and 294 are relatively prime using Euclidean algorithm.
20. Compute gcd(57, 93) and find integers x and y such that $57x + 93y = \gcd(57, 93)$.
21. Find integers x and y to satisfy: $56x + 15y = \gcd(56, 15)$.
22. Describe Extended Euclidean Algorithm. Compute multiplicative inverse of 50 in Z_{71}.
23. Find the multiplicative inverse of each nonzero element in Z_n.
24. Complete the following equalities for the numbers in GG(2):
    1+1 = ?
    1-1 = ?
    -1 = ?
    1*1 = ?
    1 * -1 = ?
25. Divide $23x^2 + 4x + 3$ by $5x + c$, assuming that the polynomials are over the field Z7.
26. What are the asymmetries between the modulo n addition and modulo n multiplication over Zn?
27. What is Euclid's algorithm for finding the GCD of two numbers? Explain.
28. Calculate the result of the following if the polynomials are over GF (2)
    $(x^4 + x^2 + x + 1) + (x^3+1)$
    $(x^4 + x^2 + x + 1) - (x^3+1)$
    $(x^4 + x^2 + x + 1) \times (x^3+1)$
    $(x^4 + x^2 + x + 1) / (x^3+1)$

❑❑❑