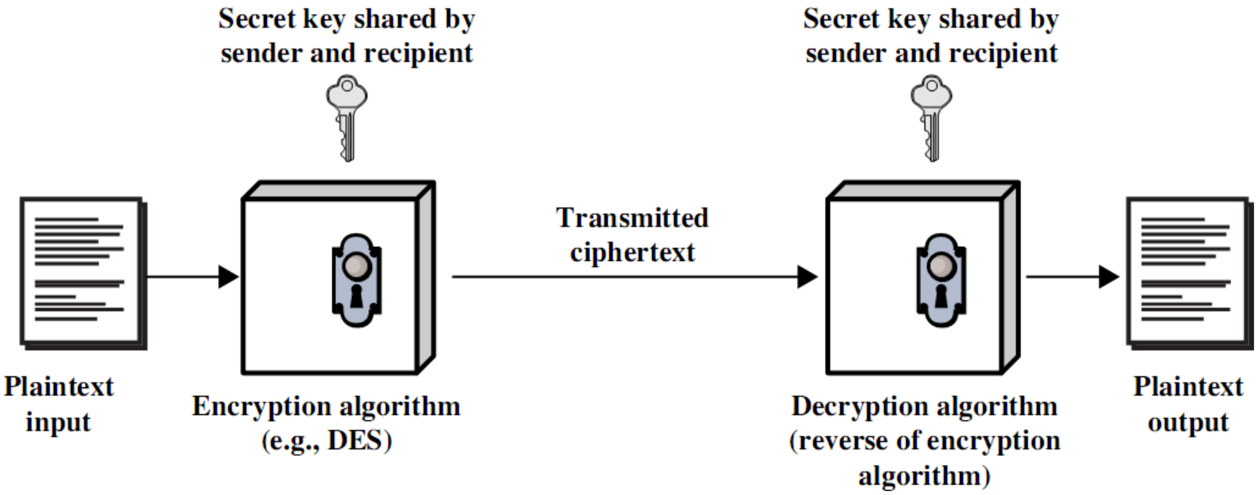
MODULE 1

CHAPTER 1: CLASSICAL ENCRYPTION TECHNIQUES

Symmetric Cipher Model:



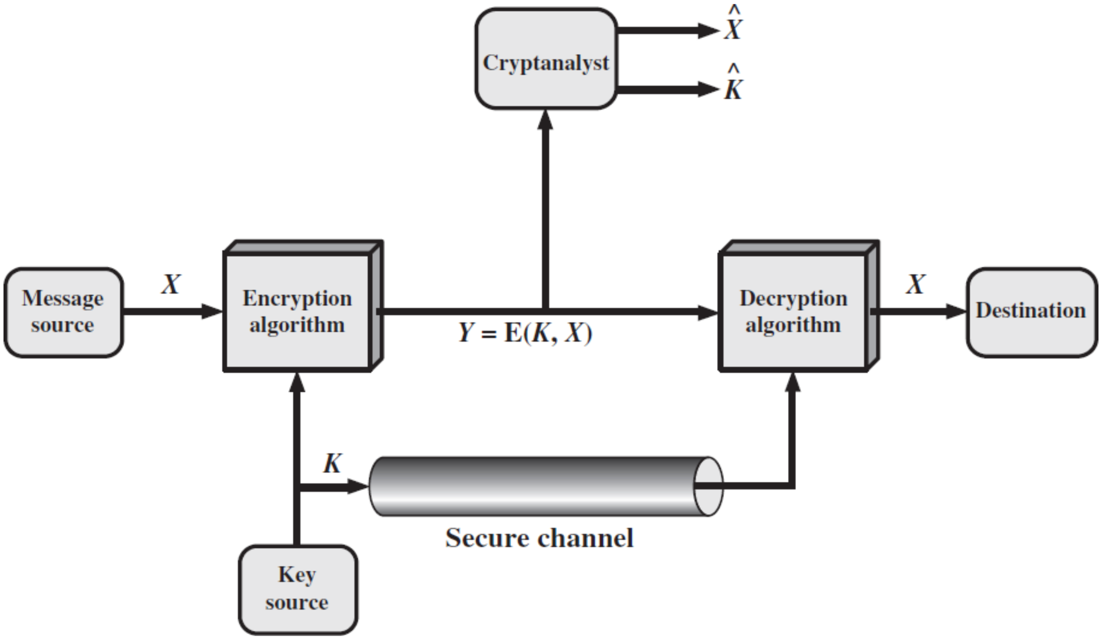
Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

A symmetric encryption scheme has five ingredients:

- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of conventional encryption:

1. **We need a strong encryption algorithm.** At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key.
2. Sender and receiver **must have obtained copies of the secret key in a secure fashion** and must keep the key secure.



A source produces a message in plaintext, $X = [X1, X2, …. XM]$. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters.

For encryption, a key of the form $K = [K1, K2, …., KJ]$ is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext $Y = [Y1, Y2, ….., YN]$. We can write this as

$$Y = \mathrm{E}(K, X)$$

This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.

The intended receiver, in possession of the key, is able to invert the transformation:

$$X = \mathrm{D}(K, Y)$$

Cryptography

Cryptographic systems are characterized along three independent dimensions:

1. **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: ***substitution,*** in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and ***transposition,*** in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost.
2. **The number of keys used.** If both sender and receiver use the same key, the system is referred to as ***symmetric, single-key, secret-key,*** or conventional encryption. If the sender and receiver use different keys, the system is referred to asasymmetric, two-key, or public-key encryption.
3. **The way in which the plaintext is processed.** A block cipher processes the input one block of elements at a time, producing an output block for each input block. A **stream cipher** processes the input elements continuously, producing output one element at a time, as it goes along.

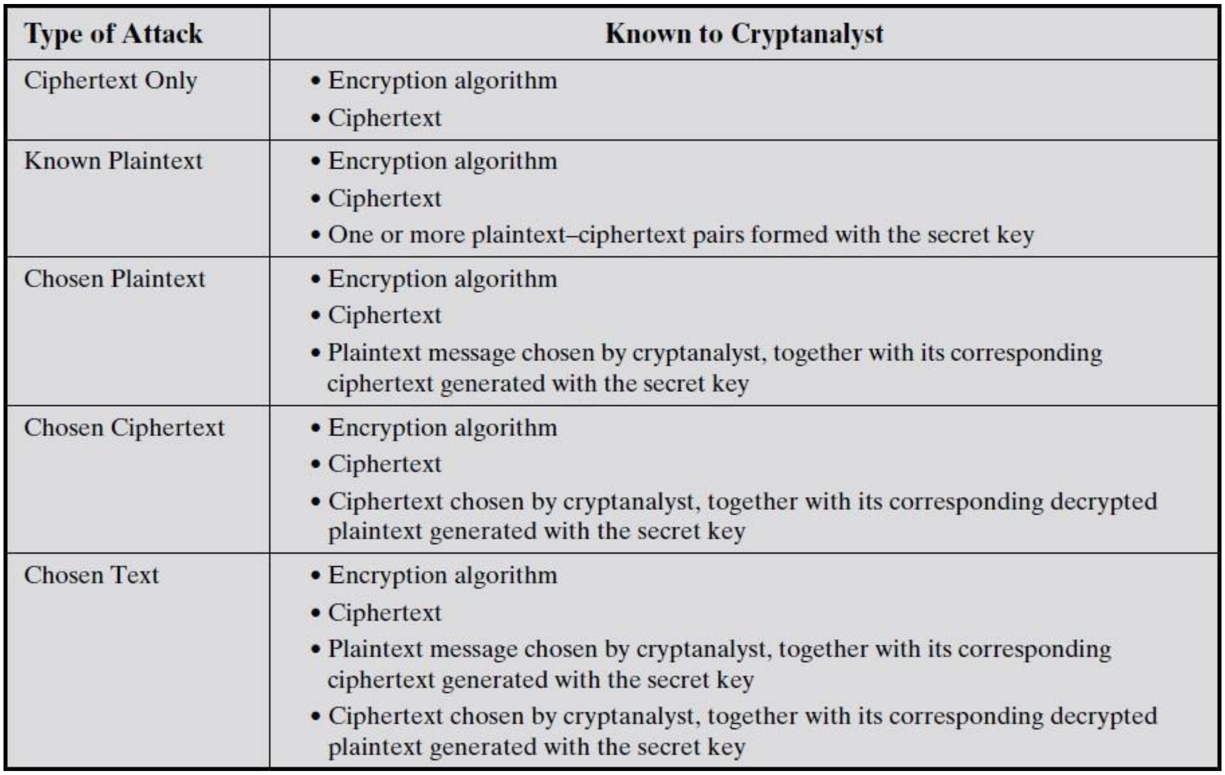
Cryptanalysis

There are two general approaches to attacking a conventional encryption scheme:

Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext.

Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

The various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst is given below.



Type of Attack	Known to Cryptanalyst
Ciphertext Only	• Encryption algorithm • Ciphertext
Known Plaintext	• Encryption algorithm • Ciphertext • One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	• Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	• Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

An encryption scheme is **unconditionally secure** if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available.

- The cost of breaking the cipher exceeds the value of the encrypted information.
- The time required to break the cipher exceeds the useful lifetime of the information.

An encryption scheme is said to be **computationally secure** if either of the foregoing two criteria are met.

Substitution Techniques:

The two basic building blocks of all encryption techniques are substitution and transposition.

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

1. Caesar Cipher:

The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example,

plain: meet me after the toga party

cipher: PHHW PH DIWHU WKH WRJD SDUWB

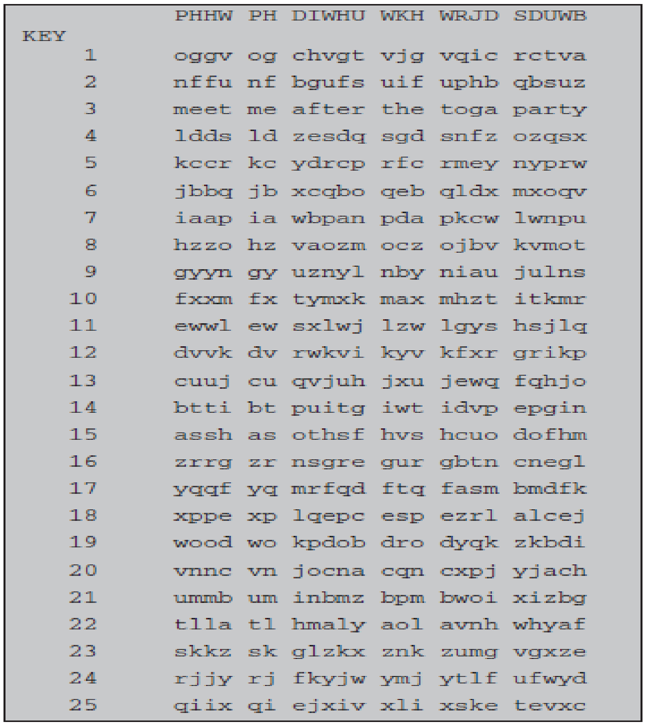
$$C = E(3,p) = (p + 3) \bmod 26$$

$$C = E(k,p) = (p + k) \bmod 26, \quad p=D(k,C) \bmod 26$$

Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.

Below Figure shows the results of applying this strategy to the example ciphertext. In this case, the plaintext leaps out as occupying the third line.



	PHHW	PH	DIWHU	WKH	WRJD	SDUWB
KEY						
1	oggv	og	chvgt	vjg	vqic	rctva
2	nffu	nf	bgufs	uif	uphb	qbsuz
3	meet	me	after	the	toga	party
4	ldds	ld	zesdq	sgd	snfz	ozqsx
5	kccr	kc	ydrcp	rfc	rmey	nyprw
6	jbbq	jb	xcqbo	qeb	qldx	mxoqv
7	iaap	ia	wbpan	pda	pkcw	lwnpu
8	hzzo	hz	vaozm	ocz	ojbv	kvmot
9	gyyn	gy	uznyl	nby	niau	julns
10	fxxm	fx	tymxk	max	mhzt	itkmr
11	ewwl	ew	sxlwj	lzw	lgys	hsjlq
12	dvvk	dv	rwkvi	kyv	kfxr	grikp
13	cuuj	cu	qvjuh	jxu	jewq	fqhjo
14	btti	bt	puitg	iwt	idvp	epgin
15	assh	as	othsf	hvs	hcuo	dofhm
16	zrrg	zr	nsgre	gur	gbtn	cnegl
17	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18	xppe	xp	lqepc	esp	ezrl	alcej
19	wood	wo	kpdob	dro	dyqk	zkbdi
20	vnnc	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzkx	znk	zumg	vgxze
24	rjjy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxc

2. Monoalphabetic Ciphers:

If the "cipher" line can be any permutation of the 26 alphabetic characters,then there are 26! possible keys. This is referred to as a monoalphabetic substitution cipher, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

If the cryptanalyst knows the nature of the plaintext (e.g., non compressed English text), then the analyst can exploit the regularities of the language. the relative frequency of the letters can be determined and compared to a standard frequency distribution for English

3. Playfair Cipher:

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword.

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

- Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that ***balloon*** would be treated as ***ba lx lo on***.
- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, **ar is encrypted as RM.**

- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. **mu is encrypted as CM.**
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP andea becomes IM (or JM, as the encipherer wishes).

4. Hill Cipher

This encryption algorithm takes *m* successive plaintext letters and substitutes for them *m* ciphertext letters. The substitution is determined by *m* linear equations in which each character is assigned a numerical value (a = 0, b = 1, c, z = 25). For *m* = 3, the system can be described as

$c1 = (k11p1 + k21p2 + k31p3) \bmod 26$

$c2 = (k12p1 + k22p2 + k32p3) \bmod 26$

$c3 = (k13p1 + k23p2 + k33p3) \bmod 26$

This can be expressed in terms of row vectors and matrices

$$(c_1\ c_2\ c_3) = (p_1\ p_2\ p_3)\begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

or

$$\mathbf{C} = \mathbf{PK} \bmod 26$$

where C and P are row vectors of length 3 representing the plaintext and ciphertext, and K is a 3 * 3 matrix representing the encryption key. Operations are performed mod 26.

For example, consider the plaintext "paymoremoney" and use the encryption key

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The first three letters of the plaintext "pay" are represented by the vector (15 0 24).

C = PK mod 26

C = (15 0 24) $\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$ mod 26

C = (303 303 531) mod 26

C = (17 17 11)

C = RRL

Same procedure is repeated for next set of letters “mor” , “emo”, “ney”. We get the cipher texts as “MWB”, “KAS”, “PDH”

So Plaintext “paymoremoney” is encrypted as RRLMWBKASPDH

Decryption requires using the inverse of the matrix K.

P = CK^{-1} mod 26

Polyalphabetic Ciphers

One of the way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message.

The general name for this approach is polyalphabetic substitution cipher. All these techniques have the following features in common:

1. A set of related monoalphabetic substitution rules is used.
2. A key determines which particular rule is chosen for a given transformation.

Vigenere Cipher

One of the simplest, polyalphabetic ciphers is the Vigenère cipher.

A general equation of the encryption process is

$$C_i = (p_i + k_{i \bmod m}) \bmod 26$$

Similarly, decryption is a generalization of Equation

$$p_i = (C_i - k_{i \bmod m}) \bmod 26$$

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is deceptive, the message "we are discovered save yourself" is encrypted as follows:

key: deceptivedeceptivedeceptive
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Decryption is equally simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column.

The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus, the letter frequency information is obscured.

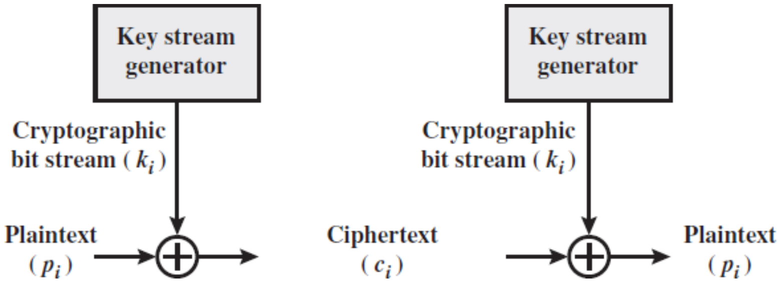
The periodic nature of the keyword can be eliminated by using a nonrepeating keyword that is as long as the message itself. Vigenère proposed what is referred to as an autokey system, in which a keyword is concatenated with the plaintext itself to provide a running key. For our example,

key: deceptivewearediscoveredsav
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

Even this scheme is vulnerable to cryptanalysis. Because the key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied.

<u>Vernam Cipher</u>

The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918.



The system can be expressed succinctly as follows

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation

Thus, the ciphertext is generated by performing the bitwise XOR of the plaintext and the key. Because of the properties of the XOR, decryption simply involves the same bitwise operation:

$$p_i = c_i \oplus k_i$$

One-Time Pad:

The key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad**, is unbreakable.

The one-time pad offers complete security but, in practice, has two fundamental difficulties:

1. There is the practical problem of making large quantities of random keys.
2. Even more daunting is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver.

CHAPTER 2. BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

- A **block cipher** is an encryption/decryption scheme in which a block of plaintextis treated as a whole and used 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 atime.

The Feistel Cipher

Feistel cipher 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.

Diffusion and Confusion

Diffusion is the statistical structure of the plaintext is ***dissipated*** into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the valueof many ciphertext digits; generally this is equivalent to having each ciphertext digit be affected by many plaintext digits.

Confusion seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key.

Feistel Cipher Structure

- The inputs to the encryption algorithm are a plaintext block of length 2w bits and akey K. The plaintext block is divided into two halves, L_0 and R_0.
- The two halves of the data pass through n rounds of processing and then combine toproduce the ciphertext block.
- Each round i has as inputs L_{i-1} and R_{i-1}, derived from the previous round, as well as asubkey K_i, derived from the overall K.
- In general, the subkeys K_i are different from K and from each other.

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. Following this substitution, a **permutation** is performed that consists of the interchange of the two halves of the data.

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

Block size: Larger block sizes mean greater security, but reduced encryption/decryption speed for a given algorithm.

Key size: Larger key size means greater security but may decrease encryption/decryptionspeed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion.

Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

Round function: Again, greater complexity generally means greater resistance to cryptanalysis.

There are two other considerations in the design of a Feistel cipher: •

- **Fast software encryption/decryption:** The speed of execution of the algorithmbecomes a concern.
- **Ease of analysis:** if the algorithm can be concisely and clearly explained, it iseasier to analyze that algorithm for cryptanalytic vulnerabilities

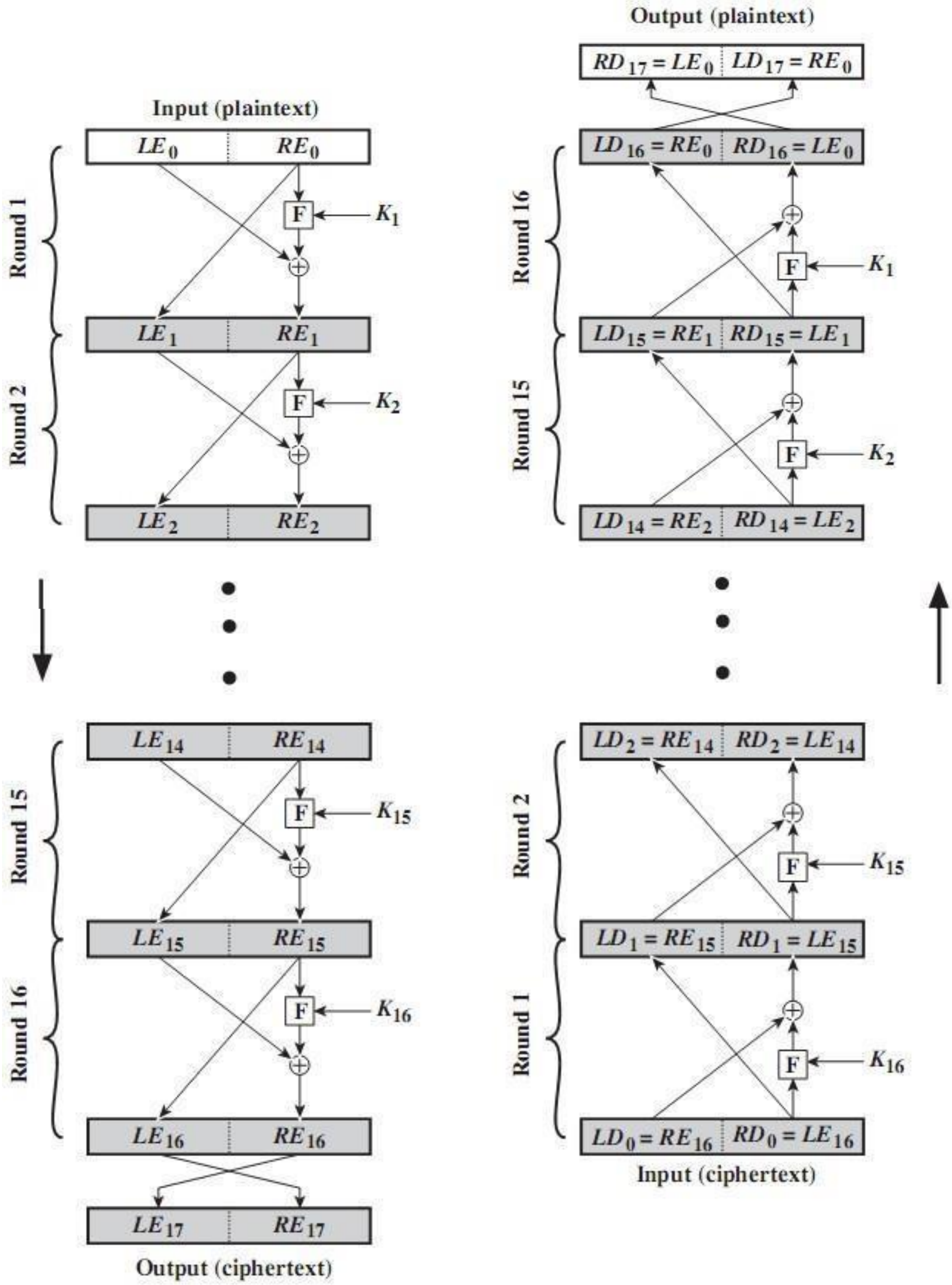


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

Feistel Decryption Algorithm

The process of decryption with a Feistel cipher is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the algorithm,but use the subkeys K_i in reverse order. That is, use K_n in the first round, K_{n-1} in the second round, and so on until K_1 is used in the last round.

Now we would like to show that the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process. First, consider the encryption process. We see that

$LE_{16} = RE_{15}$

$RE_{16} = LE_{15} \text{ x } F(RE_{15}, K_{16})$

On the decryption side, LD_1

$= RD_0 = LE_{16} = RE_{15}$

$RD_1 = LD_0 \text{ x } F(RD_0, K_{16})$

$= RE_{16} \text{ x } F(RE_{15}, K_{16})$

$= [LE_{15} \text{ x } F(RE_{15}, K_{16})] \text{ x } F(RE_{15}, K_{16})$

The XOR has the following properties:

$[A \text{ x } B] \text{ x } C = A \text{ x } [B \text{ x } C]D$

$\text{x } D = 0$

$E \text{ x } 0 = E$

Thus, we have $LD_1 = RE_{15}$ and $RD_1 = LE_{15}$. Therefore, the output of the firstround of the decryption process is $LE_{15}||RE_{15}$, which is the 32-bit swap of the input to the sixteenth round of the encryption. This correspondence holds all the way through the 16 iterations, as is easily shown. We can cast this process in general terms. For the ith iteration of the encryption algorithm,

$LE_i = RE_{i-1}$

$RE_i = LE_{i-1}$ x $F(RE_{i-1}, K_i)$

Rearranging terms,

$RE_{i-1} = LE_i$

$LE_{i-1} = RE_i$ x $F(RE_{i-1}, K_{i2} = RE_i$ x $F(LE_i, K_i)$

The Data Encryption Standard

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Institute of Standards and Technology (NIST).

The algorithm itself is referred to as the Data Encryption Algorithm (DEA). For DES, data are encrypted in **64-bit blocks using a 56-bit key.** The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

DES Encryption

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.

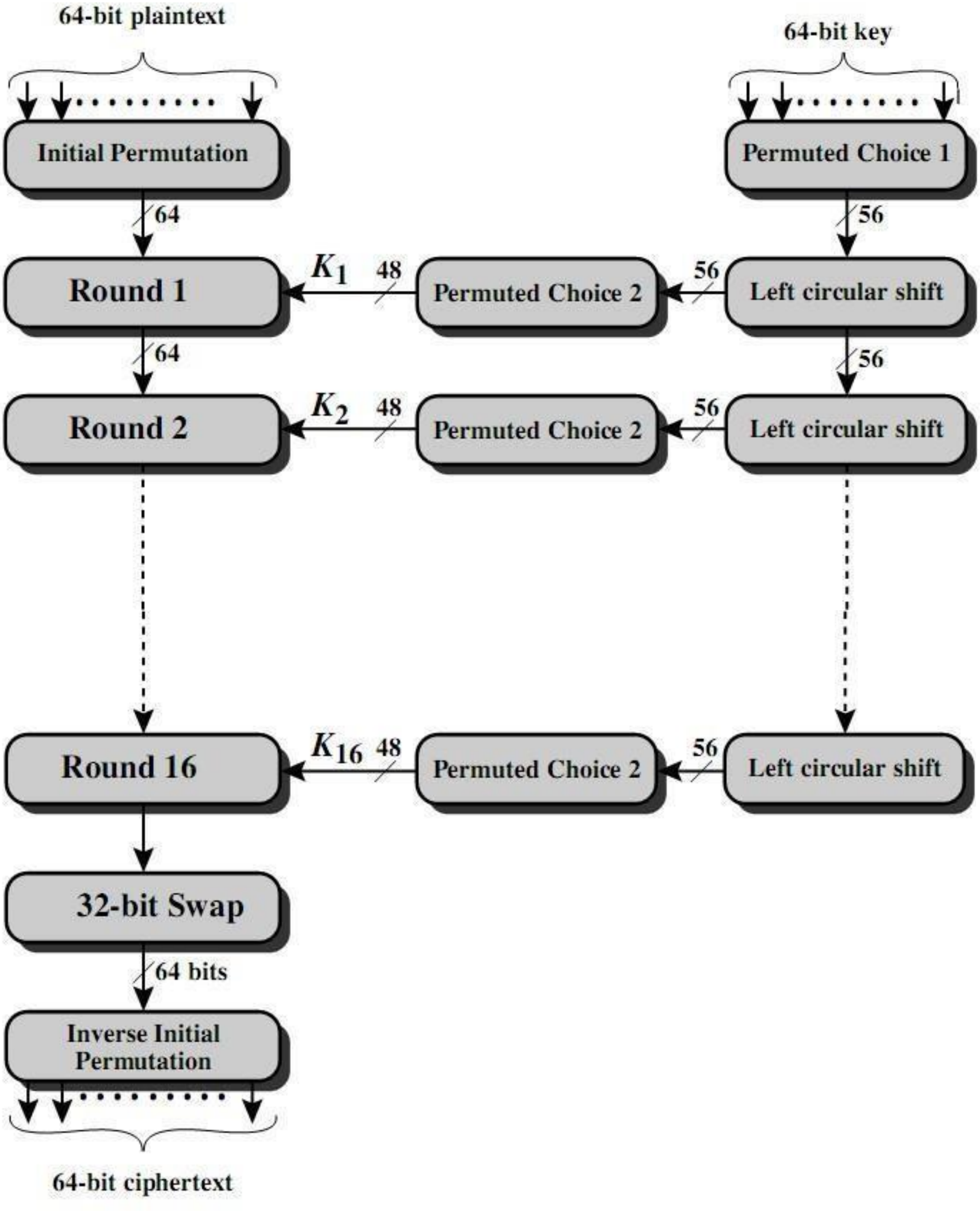


Figure 3.4 General Depiction of DES Encryption Algorithm

Looking at the left-hand side of the figure, the processing of the plaintext proceedsin three phases.

1. The 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 **16 rounds** of the same function, which involves both permutationand substitution functions.
2. The output of the last (sixteenth) round consists of 64 bits that are a function ofthe input plaintext and the key. The left and right halves of the output are **swapped** to produce the *preoutput*.
3. Finally, the preoutput is passed through a permutation **(IP^{-1})** that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher

The right-hand portion shows the way in which the 56-bit key is used. Initially, thekey is passed through a permutation function. Then, for each of the 16 rounds, a ***subkey*** (K_i) 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 subkey is produced because of the repeated shifts of the key bits.

Initial Permutation IP:

3 First step of the data computation

4 IP reorders the input data bits

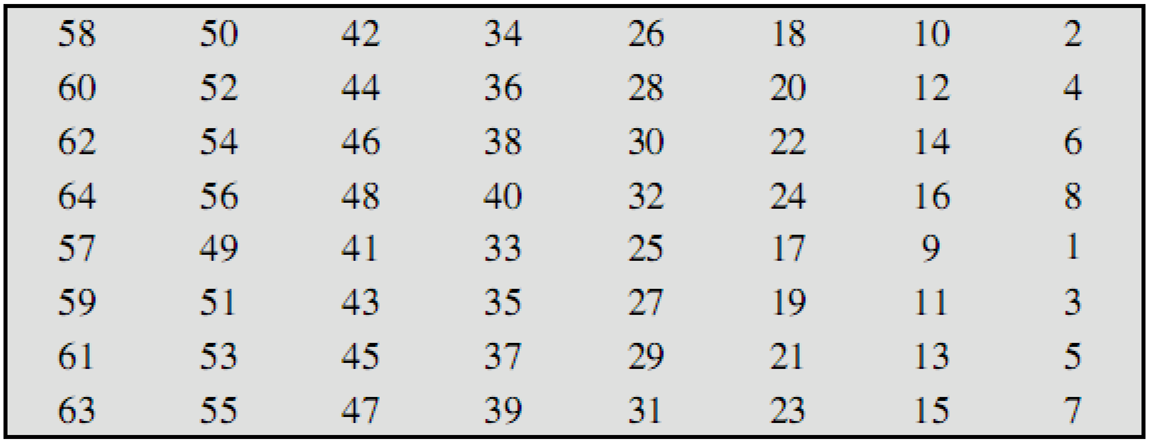
5 Even bits to LH half, Odd bits to RH half

6 Quite regular in structure (easy in h/w)

- Refer below Table for all permutation functions(IP, IP^{-1},E,P)

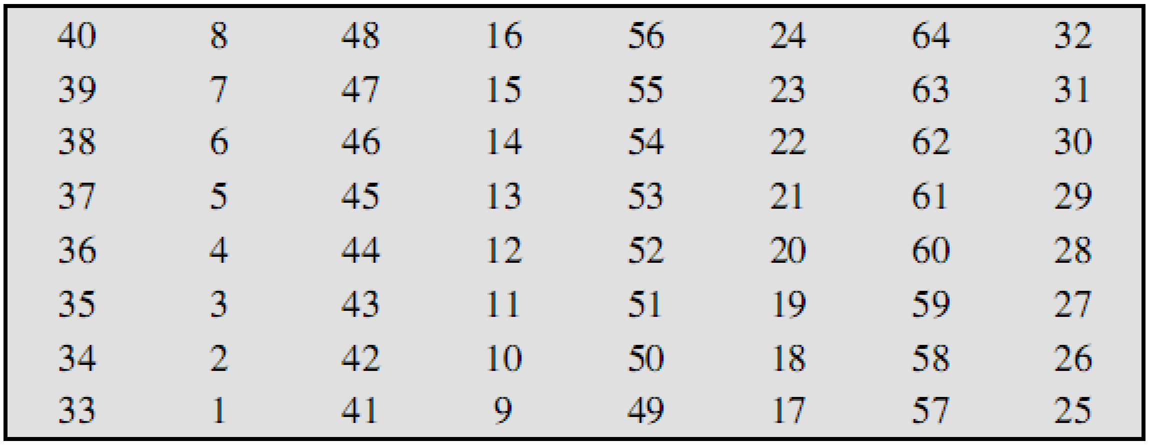
IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

(a) Initial Permutation (IP)



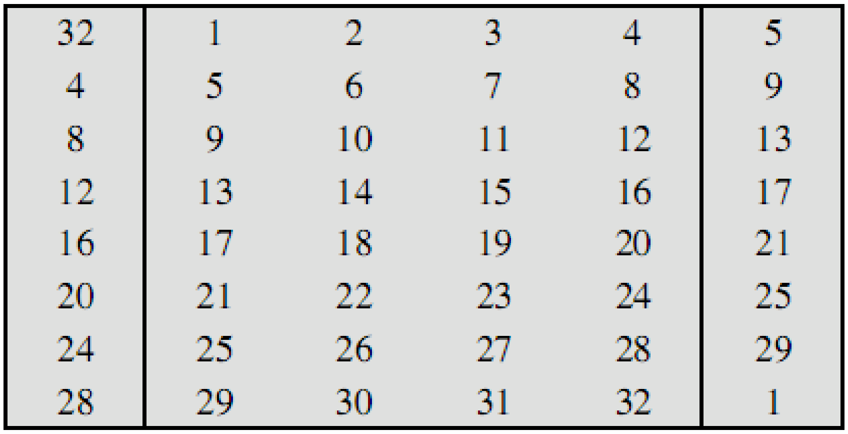
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

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



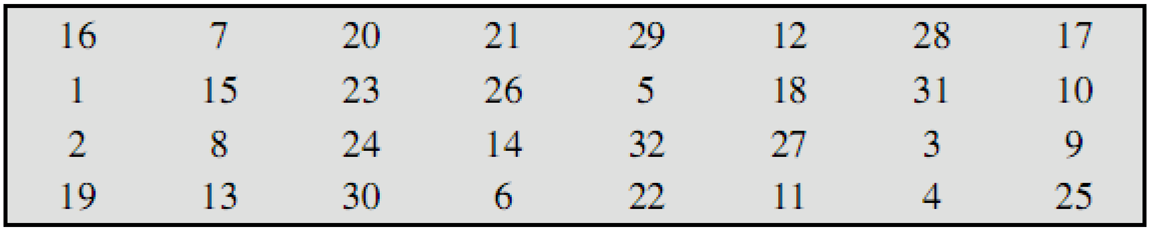
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(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

Details of Single Round:

- The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).
- the overall processing at each round can be summarized in the following formulas:

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

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

- The round key K_i 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 R bits.
- The resulting 48 bits are XORed with K_i. 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 S_i form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i. The middle four bits select one of the sixteen columns.

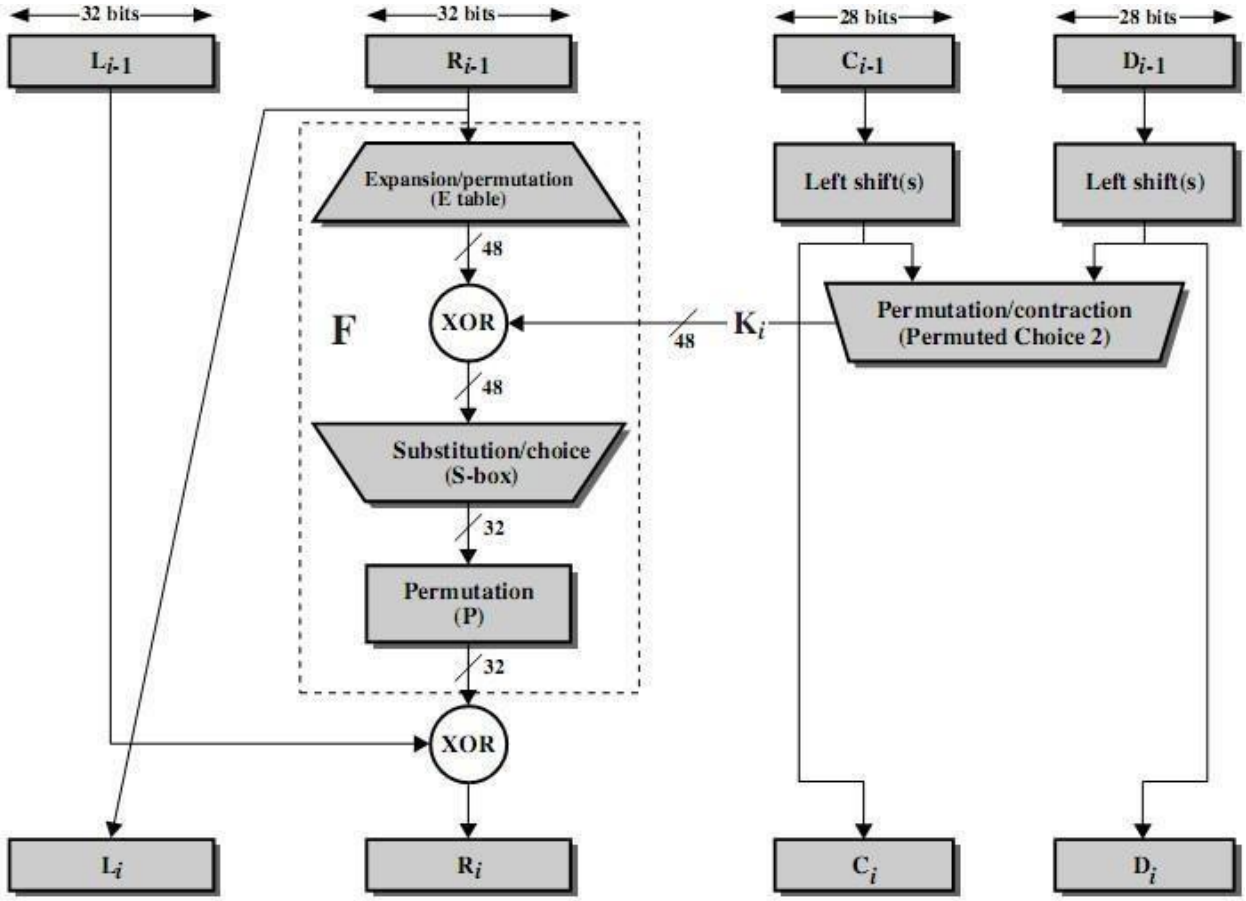
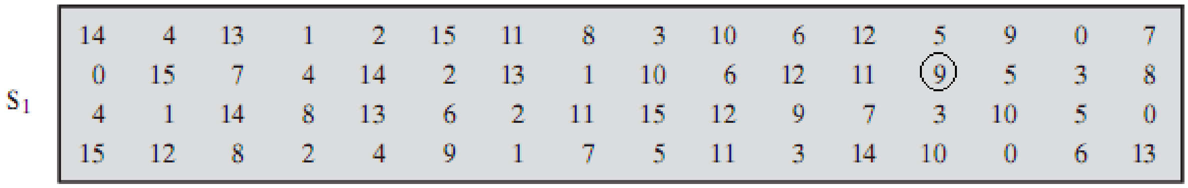


Figure 3.5 Single Round of DES Algorithm

- 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 S_1 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.

Table 3.3 Definition of DES S-Boxes



S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

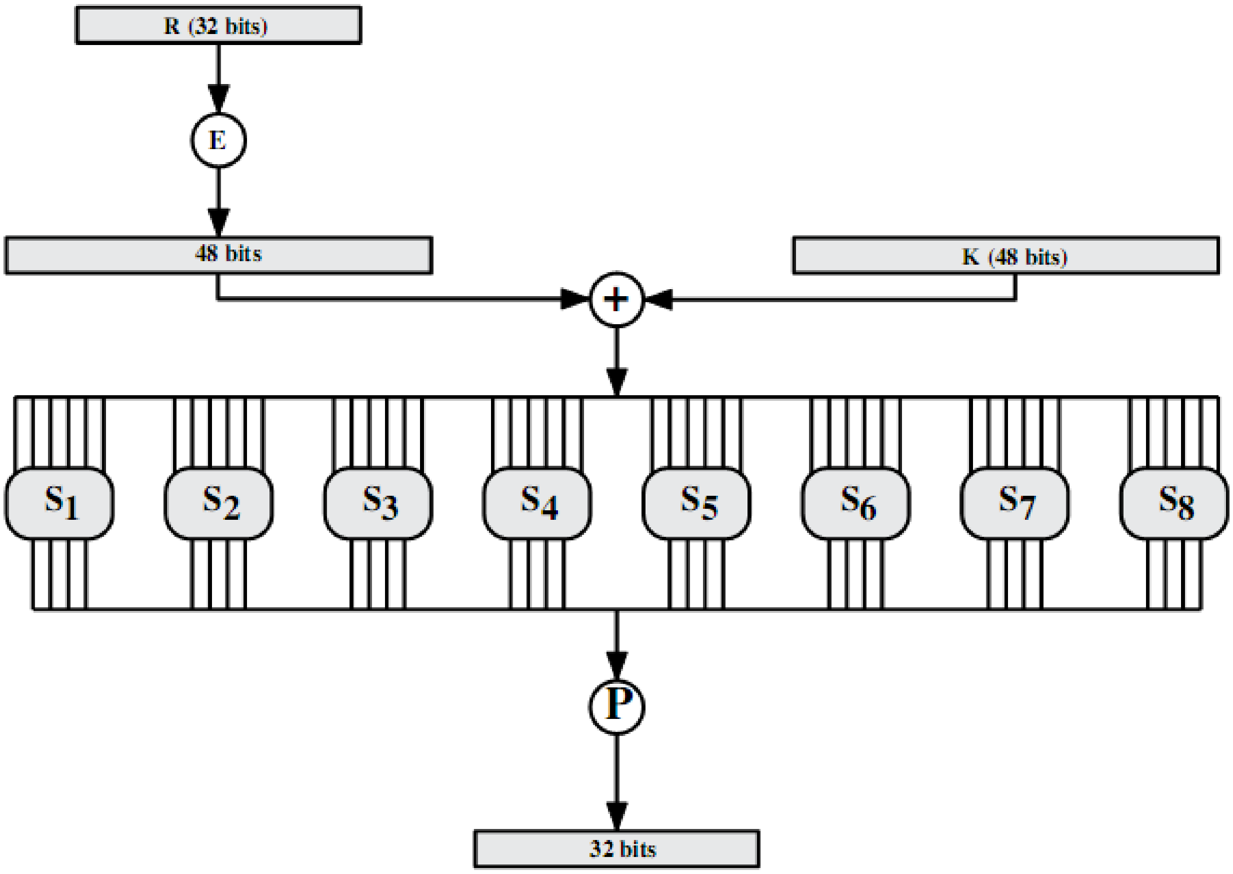


Figure 3.6 Calculation of F(R, K)

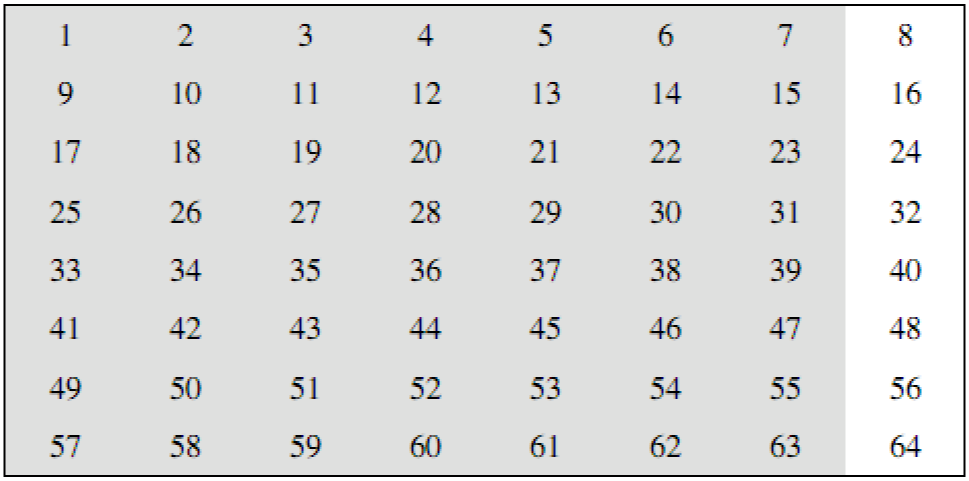
K EY GENERATION :

- Returning to Figures 3.5 and 3.6, we see that a 64-bit key is used as input tothe algorithm.
- The bits of the key are numbered from 1 through 64;every eighth bit is ignored, as indicated by the lack of shading in Table 3.4a.
- The key is first subjected to a permutation governed by a table labeled Permuted Choice One (Table 3.4b).
- The resulting 56-bit key is then treated as two 28-bit quantities, labeled C_0 and D_0.At each round, C_{i-1} and D_{i-1} are separately subjected to a circular left shift or (rotation) of 1 or 2 bits,as governed by Table 3.4d.

- These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two (Table 3.4c), which produces a 48-bit outputthat serves as input to the function F(R_{i-1},K_i).

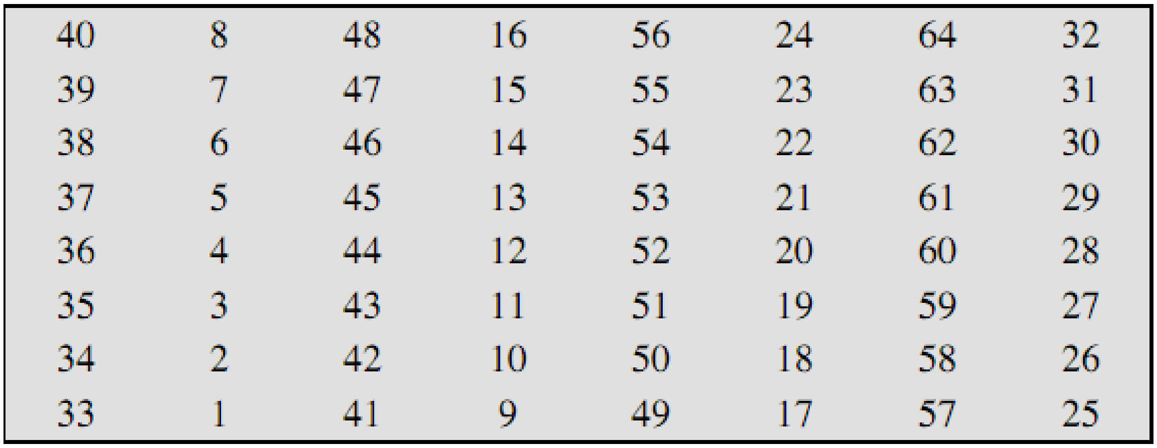
Table 3.4 DES Key Schedule Calculation

(a) Input Key



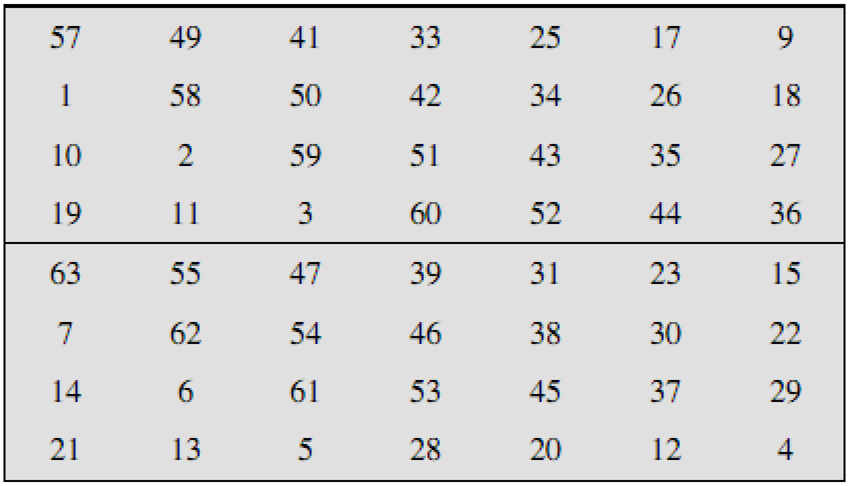
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

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



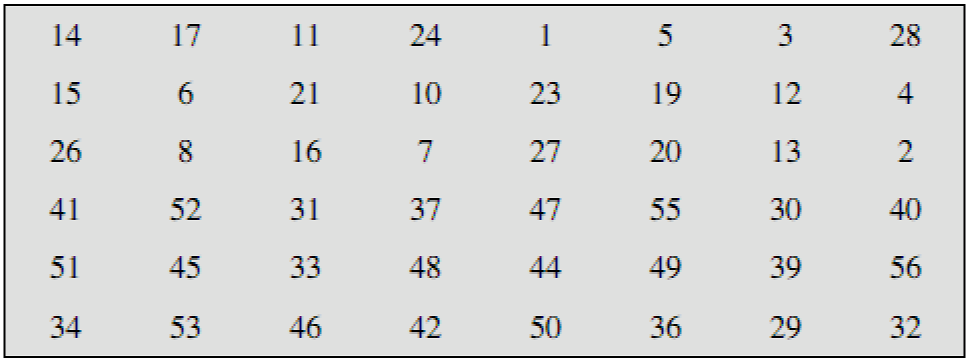
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(b) Permuted Choice One (PC-1)



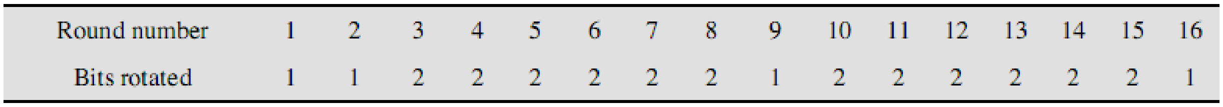
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(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

The Avalanche Effect

A change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.

The Strength of DES:

The Use of 56-Bit Keys

- With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2 x 10^{16}. So a brute-force attack appears impractical.
- Assuming that, on average, half the key space has to be searched, a singlemachine performing one DES encryption per microsecond would take more thanathousand years to break the cipher.
- If the message is just plain text in English, then the task of recognizing English would have to be automated.
- If the text message has been compressed before encryption, then recognition is more difficult. And if the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate.
- Thus, to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed.

The Nature of the DES Algorithm

Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm.

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 ciphertexts. This is a long way from knowing the actual key, but it is an intriguing first step.

Differential and Linear Cryptanalysis

Differential Cryptanalysis

One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis. In this section, we discuss the technique and its applicability to DES.The differential cryptanalysis attack is complex. 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.

Consider the original plaintext block **m** to consist of two halves $\mathbf{m_0, m_1}$. 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. If we label each new block

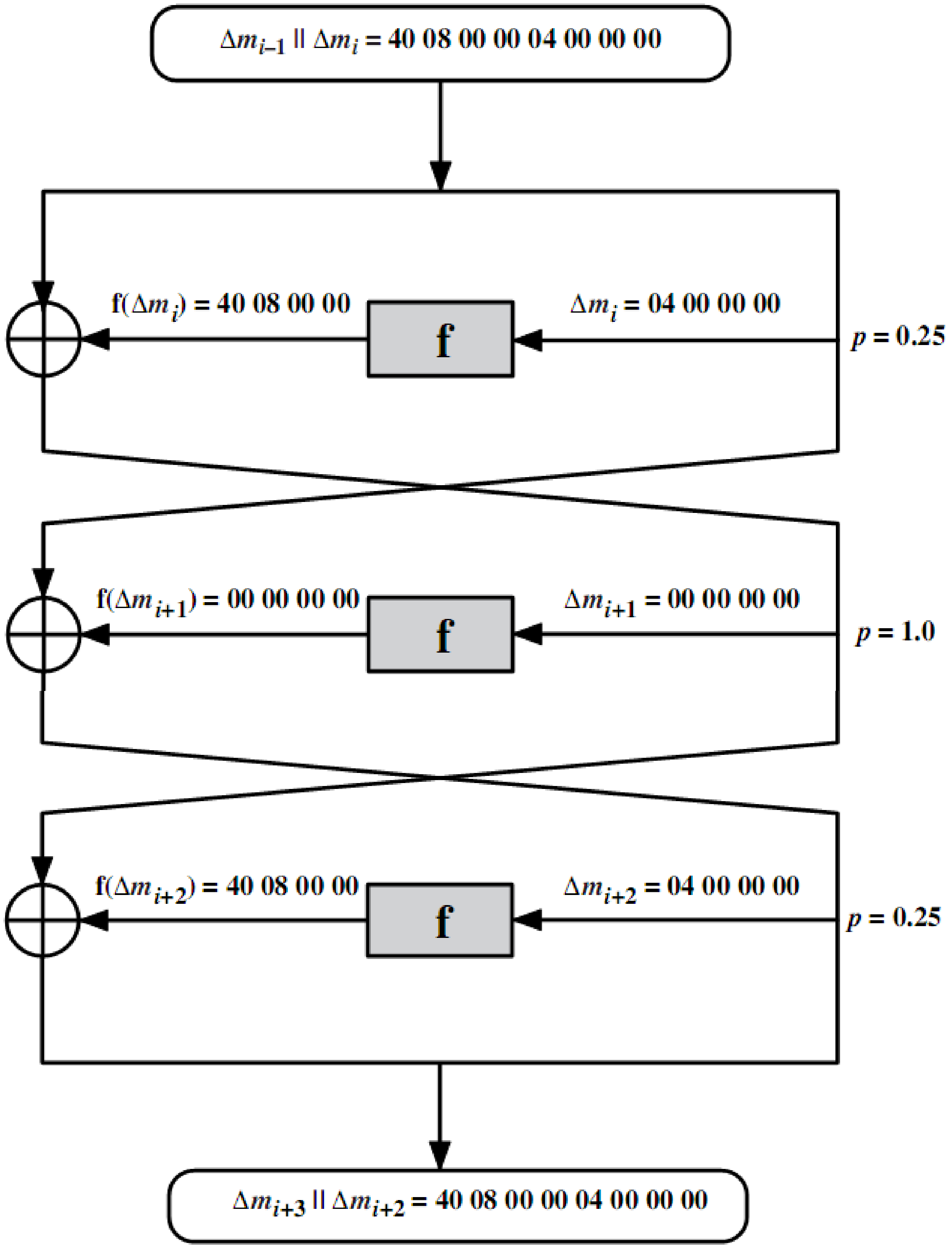
m_1 $(2 \leq i \leq 17)$, then the intermediate message halves are related as follows:

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

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

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

This attack is known as **Differential Cryptanalysis** because the analysis compares **differences** between two related encryptions, and looks for a **known difference in**leading to a **known difference out** with some (pretty small but still significant) probability. If a number of such differences are determined, it is feasible to determine the subkey used in the function f.



$\Delta m_{i-1} \| \Delta m_i$ = 40 08 00 00 04 00 00 00
f(Δm_i) = 40 08 00 00
f
Δm_i = 04 00 00 00
p = 0.25
f(Δm_{i+1}) = 00 00 00 00
f
Δm_{i+1} = 00 00 00 00
p = 1.0
f(Δm_{i+2}) = 40 08 00 00
f
Δm_{i+2} = 04 00 00 00
p = 0.25
$\Delta m_{i+3} \| \Delta m_{i+2}$ = 40 08 00 00 04 00 00 00

Figure 3.7 Differential Propagation through Three Round of DES (numbers in hexadecimal)

The overall strategy of differential cryptanalysis is based on these considerations for a single round. The procedure is to begin with two plaintext messages m and m’ witha given difference and trace through a probable pattern of differences after each round to yield a probable difference for the ciphertext. You submit **m** and **m’** for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match, then suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, can make some deductions about the key bits. This procedure must be repeated many times to determine all the keybits.

Linear Cryptanalysis

A more recent development is 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 243 known plaintexts, as compared to 247 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. Again, this attack uses structure not seen before. So far, little work has been done by other groups to validate the linear cryptanalytic approach.

Block Cipher Design Principles

There are three critical aspects of block cipher design:

- The number of rounds,
- Design of the function F,
- Key scheduling.

The number of rounds

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- 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.
- If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than brute-force key search.

Design of the function F

- The function F provides the element of confusion in a Feistel cipher, want it to be difficult to "unscramble" the substitution performed by F.
- One obvious criterion is that F be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be.
- One of the most intense areas of research in the field of symmetric block ciphers isthat of S-box design. Would like any change to the input vector to an S-box to result in random-looking changes to the output. The relationship should be nonlinear and difficult to approximate with linear functions.

Key scheduling

- A final area of block cipher design, and one that has received less attention than S-box design, is the key schedule algorithm. With any Feistel block cipher, the key schedule is used to generate a subkey for each round.
- 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.