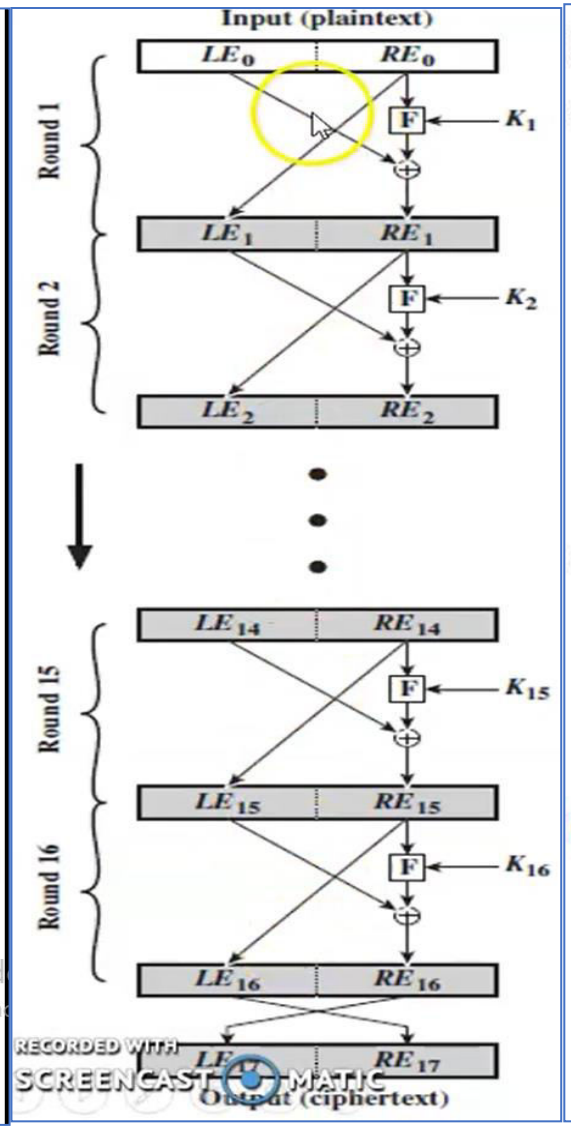
MCSE 568

Data Encryption Standard
Lecture 5

Feistel Cipher

❑ Feistel Cipher Structure

- Feistel proposed a scheme to produced a block cipher using ***permutation*** and ***substitution*** alternatively.
- The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K_i. The plaintext block is divided into two halves, LE_0 and RE_0.
- The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block.

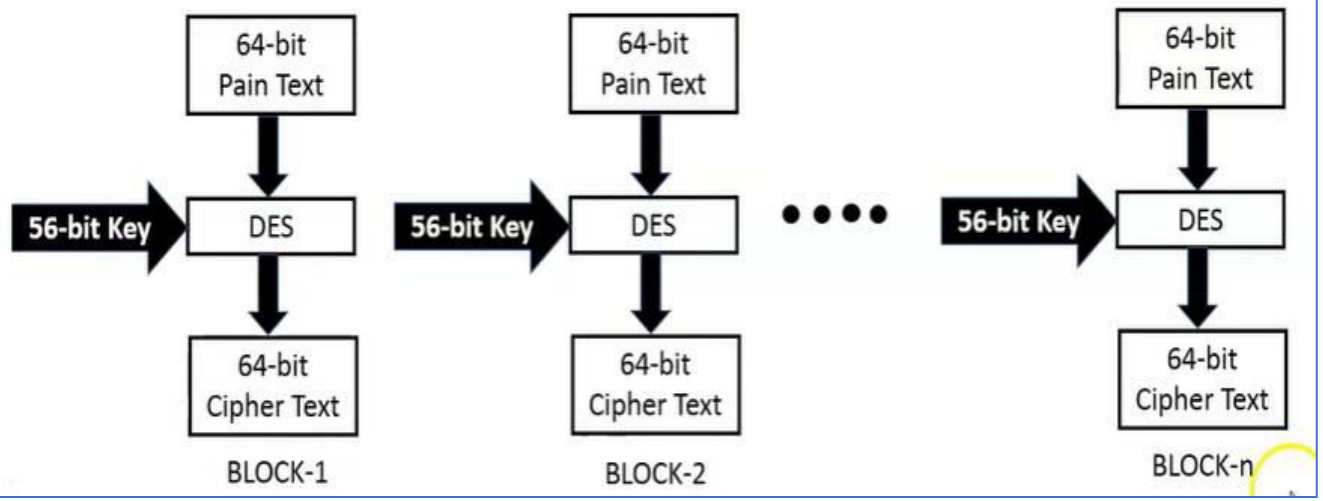


❑ Working of Feistel Cipher Structure

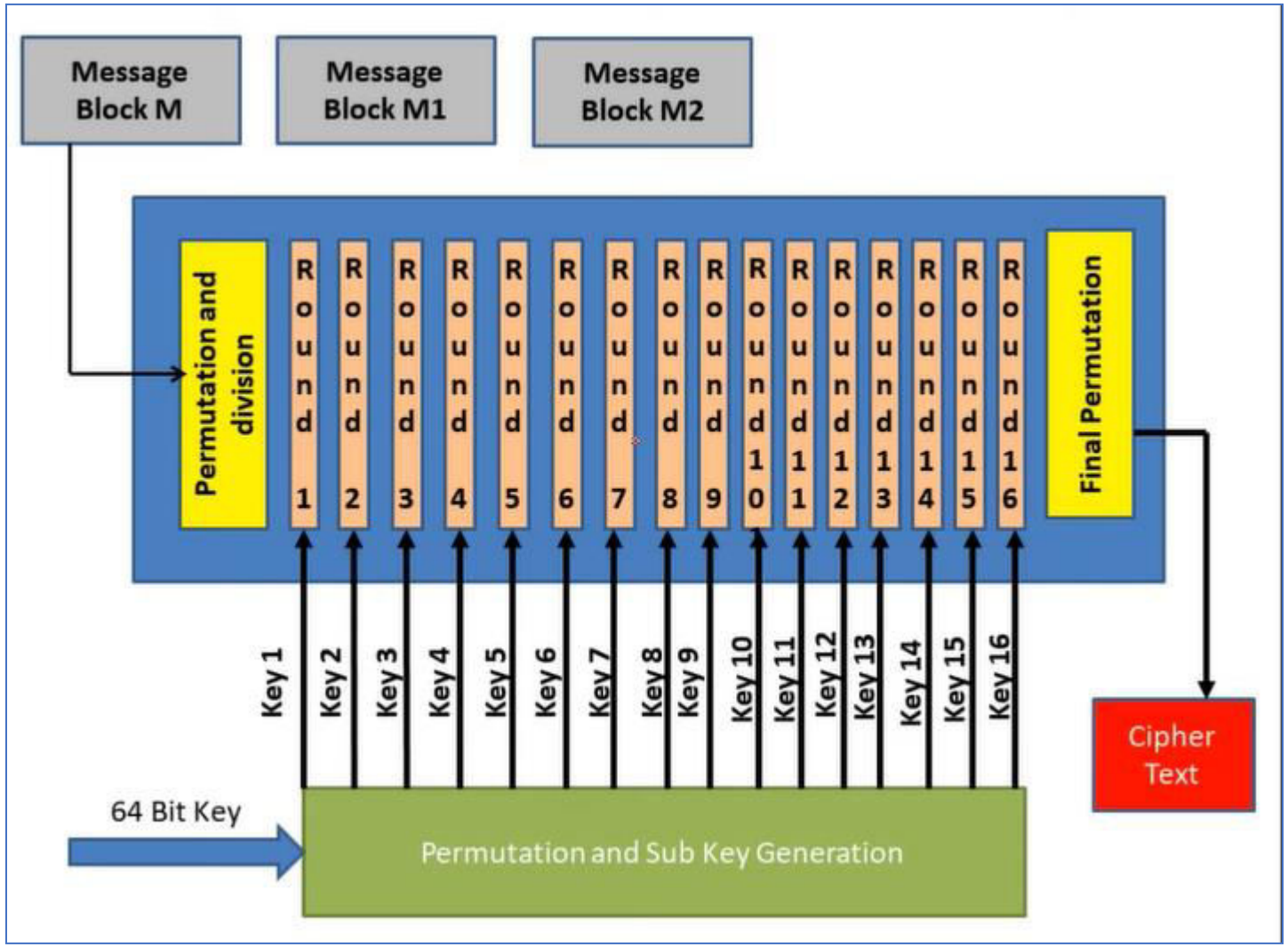
- A ***substitution*** is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data.
- The round function has the same general structure for each round but is parameterized by the round subkey K_i.
- ***Permutation*** is performed that consists of the interchange of the two halves of the data.

DES (Data Encryption Standard)

- **Introduction**
  - Developed in early 1970's at IBM and submitted to NBS .
  - DES is landmark in cryptographic algorithms.
  - DES works based on Feistel Cipher Structure.
  - DES is symmetric cipher algorithm and use block cipher method for encryption and decryption..
- **Figure shows process of DES**



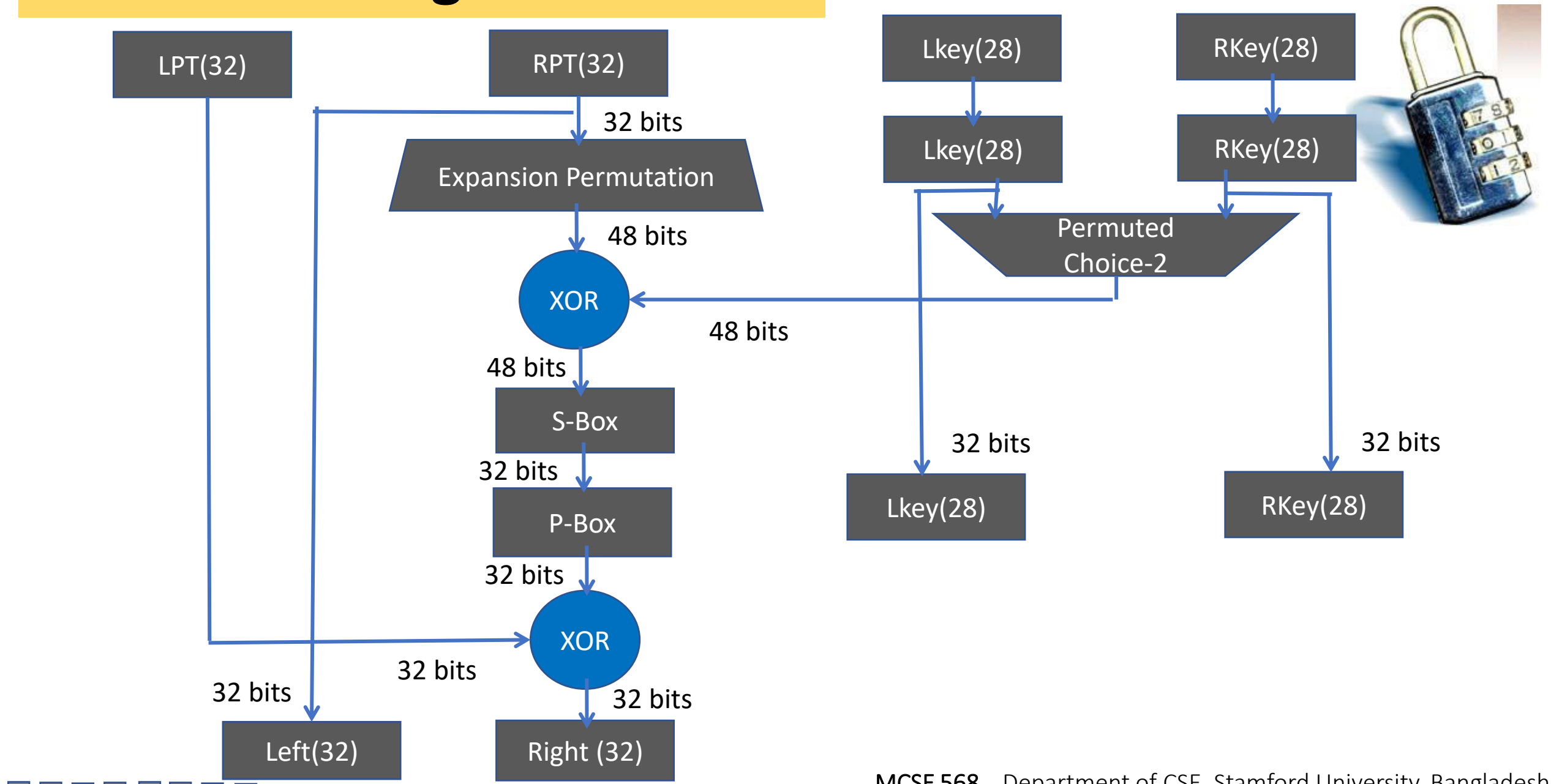
DES (Data Encryption Standard)



Phases

i) Permutation and sub key generation
ii) Plaintext permutation and division
iii) Round Functions
iv) Final Permutation

DES : Block Diagram



Step 1: Generating Sub Keys

64-bit Key=133457799BBCDFF1

Key in Hexadecimal = 133457799BBCDFF1

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	21	4

PC-1

- The 64-bit key is permuted according to the following table, **PC-1**.
- Note only 56 bits of the original key appear in the permuted key.

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

we get the 56-bit permutation
K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, C0 and D0, where each half has 28 bits.

From the permuted key K+, we get

C0 = 1111000 0110011 0010101 0101111
D0 = 0101010 1011001 1001111 0001111

From the previous slide
C0 = 1111000011001100101010101111
D0 = 0101010101100110011110001111

C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110
C_2 = 1100001100110010101010111111
D_2 = 0101010110011001111000111101
C_3 = 0000110011001010101011111111
D_3 = 0101011001100111100011110101
C_4 = 0011001100101010101111111100
D_4 = 0101100110011110001111010101
C_5 = 1100110010101010111111110000
D_5 = 0110011001111000111101010101

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

schedule of "left shifts"

C_6 = 0011001010101011111111000011
D_6 = 1001100111100011110101010101

C_7 = 1100101010101111111100001100
D_7 = 0110011110001111010101010110

C_8 = 0010101010111111110000110011
D_8 = 1001111000111101010101011001

C_9 = 0101010101111111100001100110
D_9 = 0011110001111010101010110011

C_{10} = 0101010111111110000110011001
D_{10} = 1111000111101010101011001100

C_{11} = 0101011111111000011001100101
D_{11} = 1100011110101010101100110011

C_{12} = 0101111111100001100110010101
D_{12} = 0001111010101010110011001111

C_{13} = 0111111110000110011001010101
D_{13} = 0111101010101011001100111100

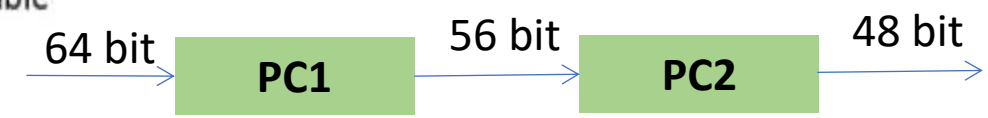
C_{14} = 1111111000011001100101010101
D_{14} = 1110101010101100110011110001

C_{15} = 1111100001100110010101010111
D_{15} = 1010101010110011001111000111

C_{16} = 1111000011001100101010101111
D_{16} = 0101010101100110011110001111

Step 1: Generating Sub Keys

We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs C_nD_n.
Each pair has 56 bits, but **PC-2** only uses 48 of these.



C_1 = 1110000110011001010101011111
D_1 = 1010101011001100111100011110

Use PC2 to calculate key k1

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

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

(1st Sub key)

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010
K_2 = 011110 011010 111011 011001 110110 111100 100111 100101
K_3 = 010101 011111 110010 001010 010000 101100 111110 011001
K_4 = 011100 101010 110111 010110 110110 110011 010100 011101
K_5 = 011111 001110 110000 000111 111010 110101 001110 101000
K_6 = 011000 111010 010100 111110 010100 000111 101100 101111
K_7 = 111011 001000 010010 110111 111101 100001 100010 111100
K_8 = 111101 111000 101000 111010 110000 010011 101111 111011
K_9 = 111000 001101 101111 101011 111011 011110 011110 000001
K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111
K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110
K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001
K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001
K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010
K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010
K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

All 16 Sub keys each of 48 bits

Step 2: Initial Permutation

Encode each 64-bit block of data.

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

There is an *initial permutation* **IP** of the 64 bits of the message data **M**

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Next divide the permuted block **IP** into a
left half L_0 of 32 bits,
and a right half R_0 of 32 bits.

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

Initial Permutation	*Final Permutation*
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

Example 6.1 Find the output of the initial permutation box when the input is given in hexadecimal as:

0x0002 0000 0000 0001

Solution The input has only two 1s (bit 15 and bit 64); the output must also have only two 1s (the nature of straight permutation). Using Table 6.1, we can find the output related to these two bits. Bit 15 in the input becomes bit 63 in the output. Bit 64 in the input becomes bit 25 in the output. So the output has only two 1s, bit 25 and bit 63. The result in hexadecimal is

0x0000 0080 0000 0002

Step 3: Round Function

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111 →32 bits

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010 →32 bits

$L_n = R_{n-1}$

$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$

Let + denote XOR addition

n=1 for round 1

$L_1 = R_{1-1}$

$R_1 = L_{1-1} \oplus f(R_{1-1}, K_1)$

$L_1 = R_0$

$R_1 = L_0 \oplus f(R_0, K_1)$

For n = 1, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010 →48 bits

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010 →32 bits

$R_1 = L_0 \oplus f(R_0, K_1)$

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

Expand 32 to 48 bit

E BIT-SELECTION TABLE

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

- To calculate f, we first expand each block R_0 from 32 bits to 48 bits.
- This is done by using a selection table that repeats some of the bits in R_0.
- We'll call the use of this selection table the function **E**.
- Thus $\mathbf{E}(R_0)$ has a 32 bit input block, and a 48 bit output block.

$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101 →48 bits

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010 →48 bits

$f(R_0,K_1) = K_1 \oplus \mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111. →48 bits

Step 3: Round Function

We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

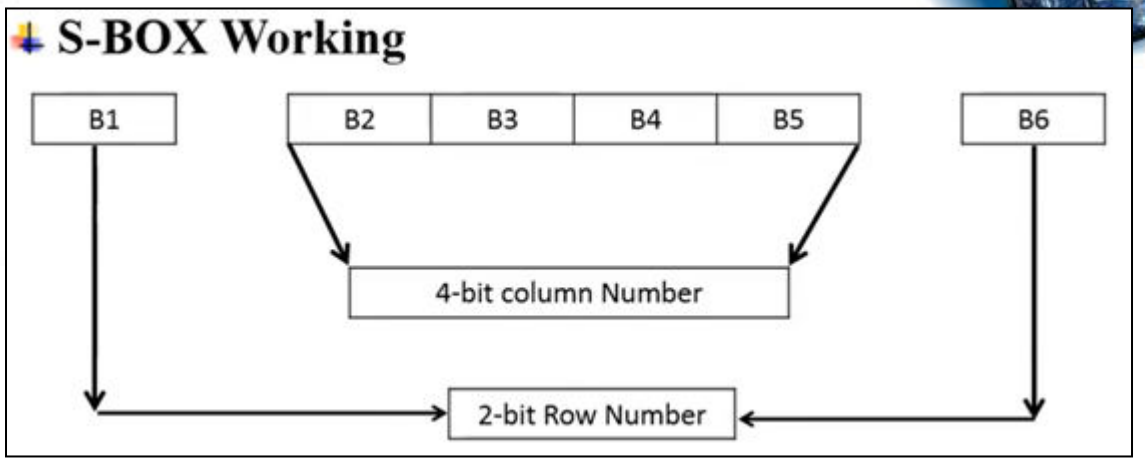
$K_1 \oplus E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111. →48 bits

$K1 \oplus E(R0)$ = $B1$ $B2$ $B3$ $B4$ $B5$ $B6$ $B7$ $B8$

We now calculate

$S_1(B_1)\ S_2(B_2)\ S_3(B_3)\ S_4(B_4)\ S_5(B_5)\ S_6(B_6)\ S_7(B_7)\ S_8(B_8)$

As L0 is 32 bits, k1+E(R0) need to compressed to 32 bits,



B_1=011000

Row No. = 00 (First bit & Last bit)

Col No. = 1100 (Middle four bits)

So $\mathbf{S_1(B_1)= 0101}$ (5)

S1

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Step 3: Round Function

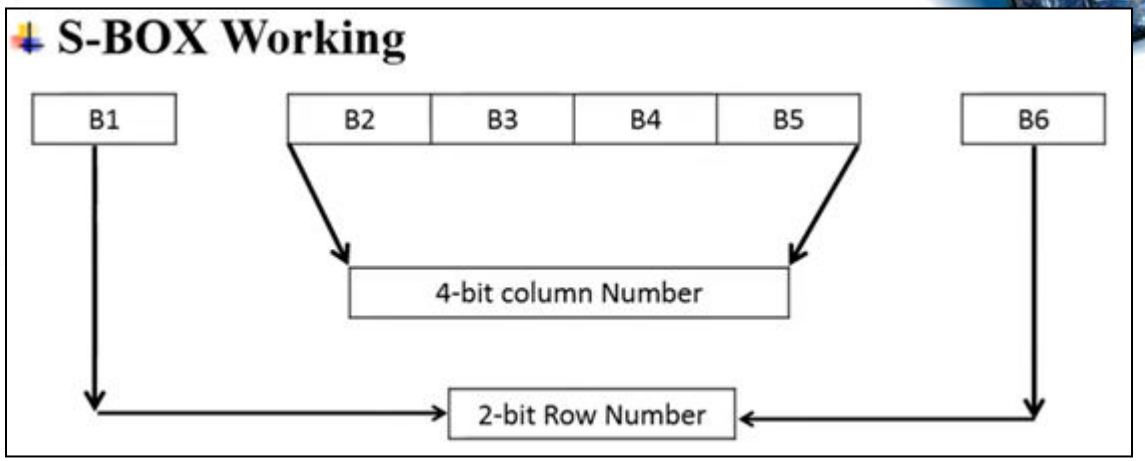
We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

$K_1+E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111. →48 bits

K1+ E(R0) = ***B1*** ***B2*** ***B3*** ***B4*** ***B5*** ***B6*** ***B7*** ***B8***

We now calculate

$S_1(B_1)$ $S_2(B_2)$ $S_3(B_3)$ $S_4(B_4)$ $S_5(B_5)$ $S_6(B_6)$ $S_7(B_7)$ $S_8(B_8)$



B_2=010001

Row No. = 01 (First bit & Last bit)

Col No. = 1000 (Middle four bits)

So **$S_2(B_2)$= 1100** (12)

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S2

Step 3: Round Function

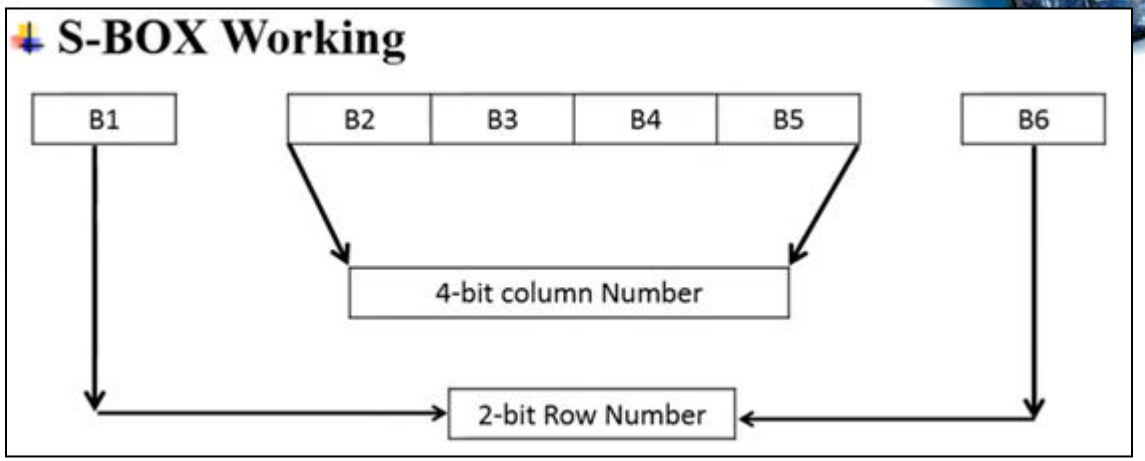
We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

$K_1+E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111. →48 bits

K1+ E(R0) = ***B1 B2 B3 B4 B5 B6 B7 B8***

We now calculate

$S_1(B_1)\ S_2(B_2)\ S_3(B_3)\ S_4(B_4)\ S_5(B_5)\ S_6(B_6)\ S_7(B_7)\ S_8(B_8)$



B_3=011110

Row No. = 00 (First bit & Last bit)

Col No. = 1111 (Middle four bits)

So **$S_3(B_3)$= 1000** (8)

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S3

Step 3: Round Function

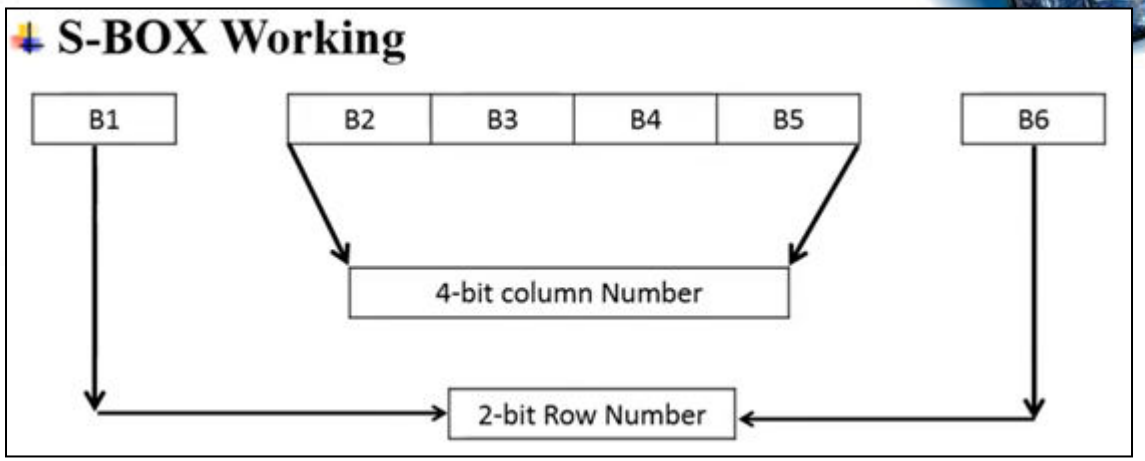
We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

$K_1+E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111. →48 bits

K1+ E(R0) = B1 B2 B3 B4 B5 B6 B7 B8

We now calculate

$S_1(B_1)\ S_2(B_2)\ S_3(B_3)\ S_4(B_4)\ S_5(B_5)\ S_6(B_6)\ S_7(B_7)\ S_8(B_8)$



B_4=111010

Row No. = 10 (First bit & Last bit)

Col No. = 1101 (Middle four bits)

So **$S_4(B_4)$= 0010** (2)

Row No.	Column Number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S4

Step 3: Round Function

We now do something strange with each group of six bits: we use them as addresses in tables called "**S boxes**".

$K_1+E(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

K1+ E(R0) = B1 B2 B3 B4 B5 B6 B7 B8

We now calculate

$S_1(B_1)\ S_2(B_2)\ S_3(B_3)\ S_4(B_4)\ S_5(B_5)\ S_6(B_6)\ S_7(B_7)\ S_8(B_8)$

$S_5(B_5)$ 100001

$S_5(B_5)$ 1011

S5

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

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

= 0101 1100 1000 0010 1011 0101 1001 0111

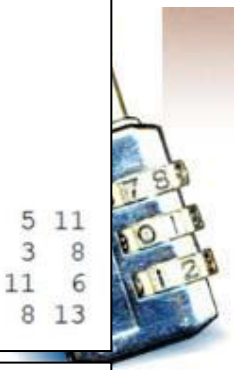
→32 bits

$S_6(B_6)$ 100110

$S_6(B_6)$ 0101

S6

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



$S_7(B_7)$ 010100

$S_7(B_7)$ 1001

S7

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

$S_8(B_8)$ 100111

$S_8(B_8)$ 0111

S8

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

Step 3: Round Function

$f(R_0,K_1)$ - Final stage of ***f- Permutation***

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

= 0101 1100 1000 0010 1011 0101 1001 0111

$f(R_0,K_1)$ = **0010 0011 0100 1010 1010 1001 1011 1011** →32 bits

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

Finding output of Round 1

L0 = 1100 1100 0000 0000 1100 1100 1111 1111 →32 bits

R0 = 1111 0000 1010 1010 1111 0000 1010 1010 →32 bits

For n = 1, we have

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010 →32 bits

$R_1 = L_0 \oplus f(R_0,K_1)$

= 1100 1100 0000 0000 1100 1100 1111 1111 →32 bits

⊕ 0010 0011 0100 1010 1010 1001 1011 1011 →32 bits

R1 = 1110 1111 0100 1010 0110 0101 0100 0100 →32 bits

L1 = 1111 0000 1010 1010 1111 0000 1010 1010 →32 bits

For Round 2

$L_2=R_1$

$R_2=L_1 \oplus f(R_1,K_2)$

Continue upto round 16…

Step 4: Final Permutation

Finally after 16 Rounds

Output of 16 Rounds

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100 →32 bits
R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101 →32 bits

64 bits

We reverse the order of these two blocks and apply the final permutation to
$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

64 bits

Cyphertext which in hexadecimal format is

85E813540F0AB405.

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

The Truth

- DES is insecure due to the relatively short 56-bit key size.
- In January 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes
- This cipher has been superseded by the Advanced Encryption Standard (AES).
- DES has been withdrawn as a standard by the National Institute of Standards and Technology.

Same procedure must be repeated for other blocks

Check the result online using https://emvlab.org/descalc/

DES (Data Encryption Standard)

Example 6.3 The input to S-box 1 is 100011. What is the output?

Solution If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

Example 6.4 The input to S-box 8 is 000000. What is the output?

Solution If we write the first and the sixth bits together, we get 00 in binary, which is 0 in decimal. The remaining bits are 0000 in binary, which is 0 in decimal. We look for the value in row 0, column 0, in Table 6.10 (S-box 8). The result is 13 in decimal, which is 1101 in binary. So the input 000000 yields the output 1101.

DES (Data Encryption Standard)

Example 6.5 We choose a random plaintext block and a random key, and determine what the ciphertext block would be (all in hexadecimal):

Plaintext: 123456ABCD132536 Key: AABB09182736CCDD
CipherText: C0B7A8D05F3A829C

Let us show the result of each round and the text created before and after the rounds. Table 6.15 first shows the result of steps before starting the round.

Table 6.15 *Trace of data for Example 6.5*

Plaintext: 123456ABCD132536			
After initial permutation: 14A7D67818CA18AD			
After splitting: L_0=14A7D678 R_0=18CA18AD			
Round	*Left*	*Right*	*Round Key*
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3
Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC

Table 6.15 (Contd.)

Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C			*(after final permutation)*

The plaintext goes through the initial permutation to create completely different 64 bits (16 hexadecimal digit). After this step, the text is split into two halves, which we call L_0 and R_0. The table shows the result of 16 rounds that involve mixing and swapping (except for the last round). The results of the last rounds (L_{16} and R_{16}) are combined. Finally the text goes through final permutation to create the ciphertext.

Q1: The DES Algorithm Cipher System consists of _____rounds (iterations)

A 12

B 16

C 18

D 8

30

Q2: The DES algorithm has a key length of __

A 32 bits

B 48 bits

C 56 bits

D 64 bits

Q3: The input and output of S-Box are _____

A. 48 & 32 bits

B. 32 & 48 bits

C. 56 & 32 bits

D. 32 & 32 bits

Q4: The Initial Permutation table/matrix is of size _____

A 8x4

B 12x8

C 8x8

D 16x8

Q5: The number of tests required to break the DES algorithm is_____

A 2.8×10^{14}

B 4.2×10^{9}

C 1.84×10^{19}

D 7.2×10^{16}