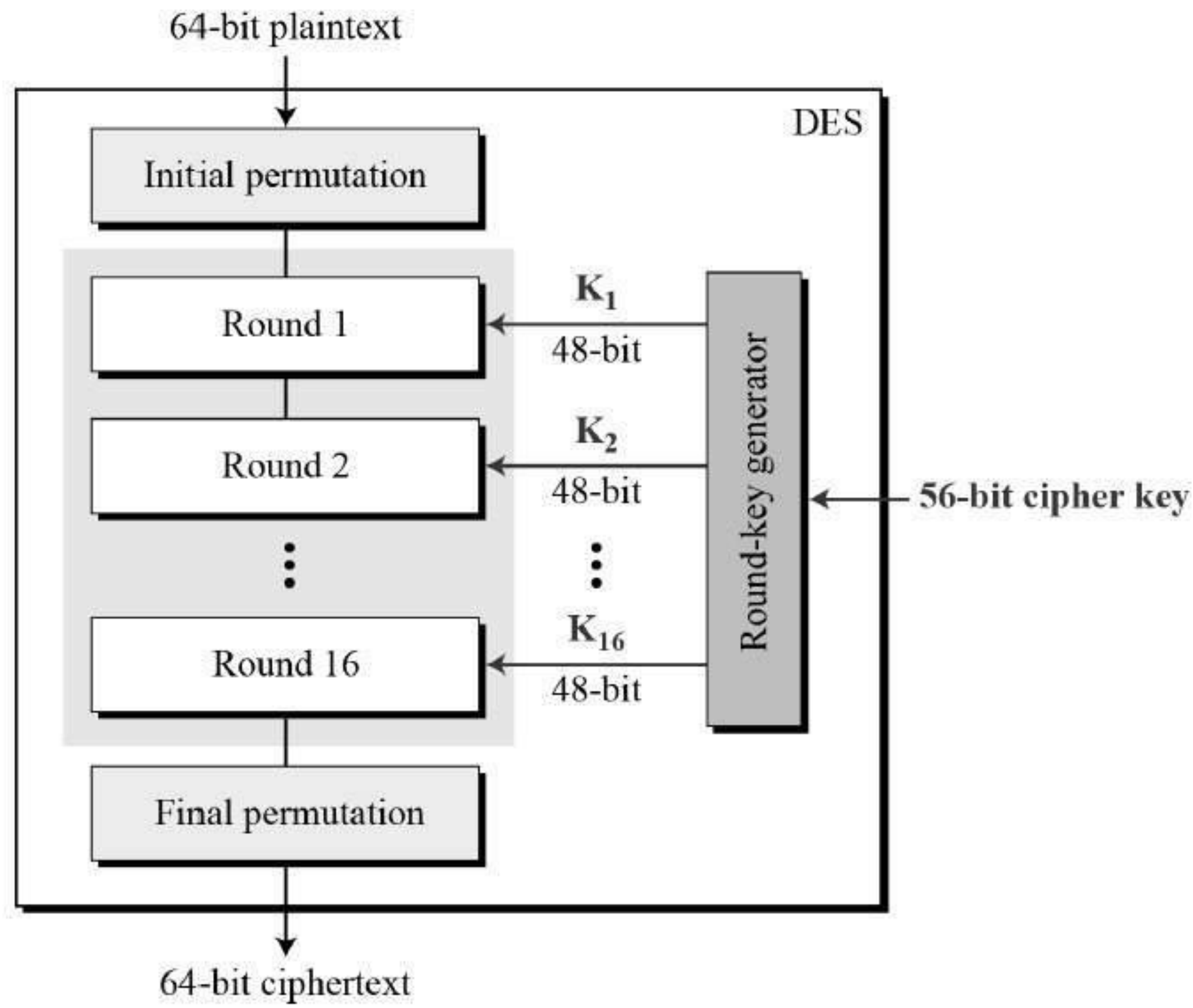
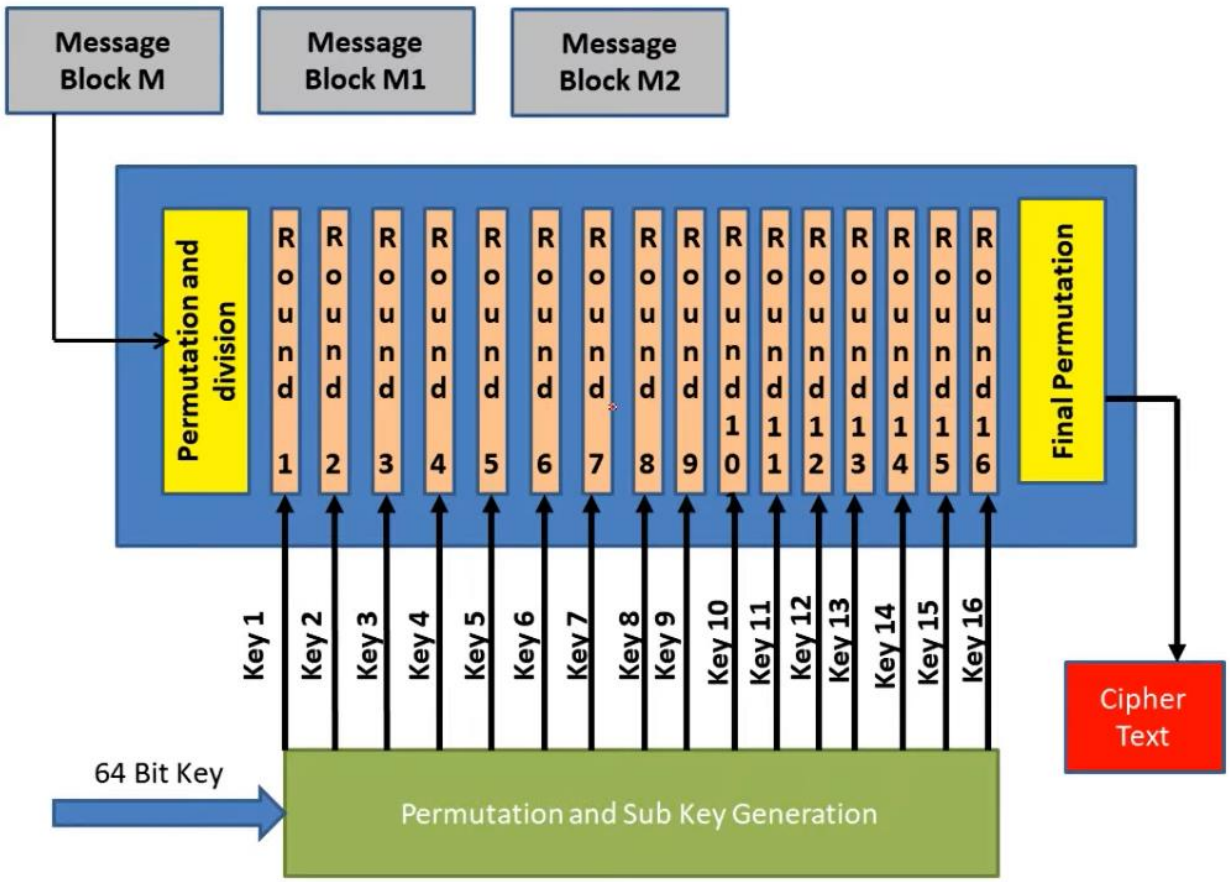
Data Encryption Standard (DES)

- Data Encryption Standard (DES) refers to a symmetric key encryption algorithm that was developed by the National Institute of Standards and Technology (NIST) in 1977 as an official standard for encrypting data.
- In symmetric-key algorithms, the same keys are used for both encryption and decryption thus, secure communication heavily relies on proper key management.
- DES is an implementation of a Feistel Cipher. It uses 16 rounds where substitution and permutation steps are applied to convert plaintext into ciphertext during each round of processing done by the algorithm. The plain text block size is 64-bit.
- Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).
- Although AES has replaced it with stronger methods such as triple DES (3DES), some old systems still use this technology because it remains significant within cryptographic history.
- General Structure of DES is depicted in the following illustration



Since DES is based on the Feistel Cipher, all that is required to specify DES is –

- Round function
- Key schedule
- Any additional processing − Initial and final permutation



Phase 1

Generating 16 Sub keys

Key in Hexadecimal = 133457799BBCDFF1

K= 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

- A **64-bit key** is taken as input.
- This key is **permuted using the PC-1 table** (Permutation Choice 1).
- Only **56 bits** are selected; the remaining 8 bits (typically parity bits) are discarded.

PC-1 Table:

This table specifies the **bit positions** from the original 64-bit key that will be used (note: 8 parity bits are discarded).

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

we get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 00011111

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

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

Schedules of "Left shift"

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

Here we should apply circular left shift on C0 and D0 according to given schedules of left sift table.

C1 = 1110000110011001010101011111
D1 = 1010101011001 100111100011110

C2 = 1100001100110010101010111111
D2 = 010101011001 1001111000111101

C3 = 0000110011001010101011111111
D3 = 0101011001 100111100011110101

C4 =
D4 =

C5 =
D5 =

C6 =
D6 =

C7 =
D7 =

C8 =
D8 =

C9 =
D9 =

C10 =
D10 =

C11 =
D11 =

C12 =
D12 =

C13 =
D13 =

C14 =
D14 =

C15 =
D15 =

C16 =
D16 =

We now form the keys Kn, for 1≤n≤16, by applying the following permutation table to each of the concatenated pairs CnDn.

Each pair has 56 bits, but PC-2 only uses 48 of these.

So, we are going to take C1D1 and apply on PC-2

C1 = 1110000110011001010101011111
D1 = 1010101011001 100111100011110

C1D1=11100001100110010101010111111010101011001100111100011110

PC-2 Table:

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

After applying PC-2 on C1D1 we get Key K1

K1 = 000110 110000 001011 101111 111111 000111 000001 110010

C2 = 1100001100110010101010111111

D2 = 010101011001 1001111000111101

C2D2= 1100001100110010101010111111010101011001 1001111000111101

After applying PC-2 on C2D2 we get Key K2

K2=011110011010111011011001110110111100100111100101

So all 16 keys are as follows

K1=000110110000001011101111111111000111000001110010

K2=011110011010111011011001110110111100100111100101

K3=010101011111110010001010010000101100111110011001

K4=011100101010110111010110110110011010100011101

K5=011111001110110000000111111010110101001110101000

K6=011000111010010100111110010100000111101100101111

K7=111011001000010010110111111101100001100010111100

K8=111101111000101000111010110000010011101111111011

K9=111000001101101111101011111011011110011110000001

K10=101100011111001101000111101110100100011001001111

K11=001000010101111111010011110111101101001110000110

K12=011101010111000111110101100101000110011111101001

K13=100101111100010111010001111110101011101001000001

K14=010111110100001110110111111100101110011100111010

K15=101111111001000110001101001111010011111100001010

K16=110010110011110110001011000011100001011111110101

Phase 2

Step-2: Encode each 64-bit block of data

Let's suppose Message is

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 by applying IP table

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

we get

IP=1100110000000000110011001111111111110000101010101111000010101010

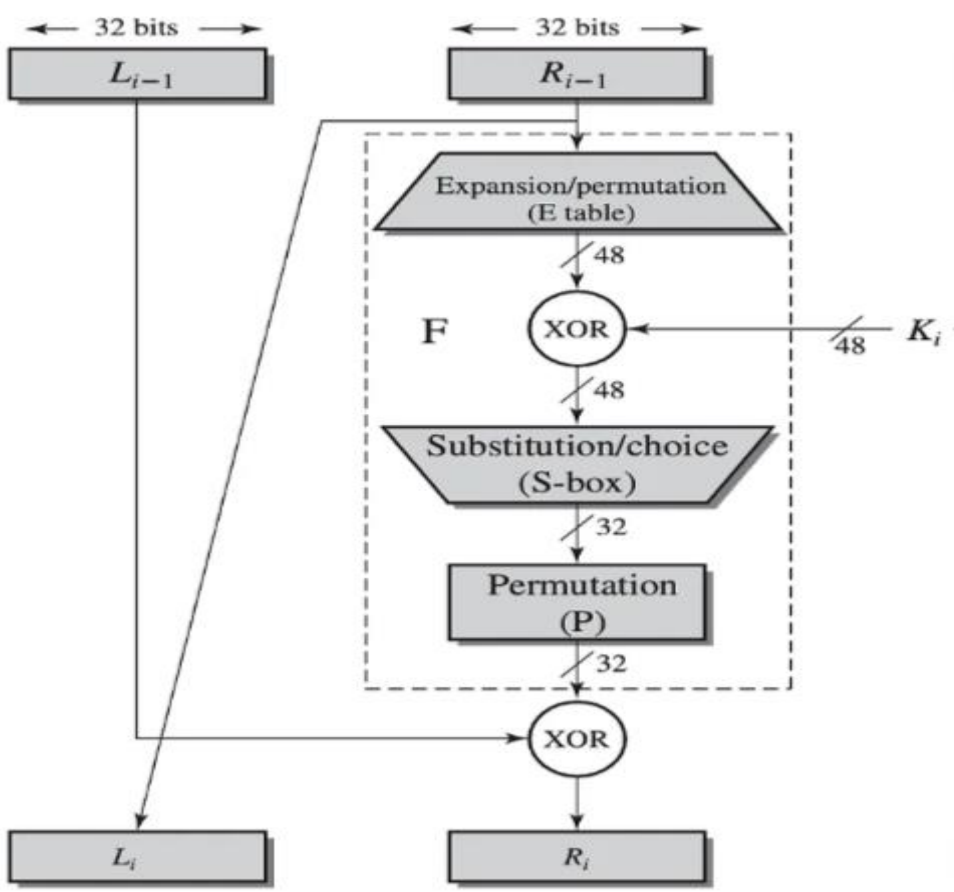
"Next divide the permuted block IP into a
left half L0 of 32 bits,
and a right half R0 of 32 bits."
L0=11001100000000001100110011111111

R0=11110000101010101111000010101010

Phase 3

What Happens in Every Single Round?

- In a single round, the total input size is 64-bit into two halves 32–32-bit. The figure is

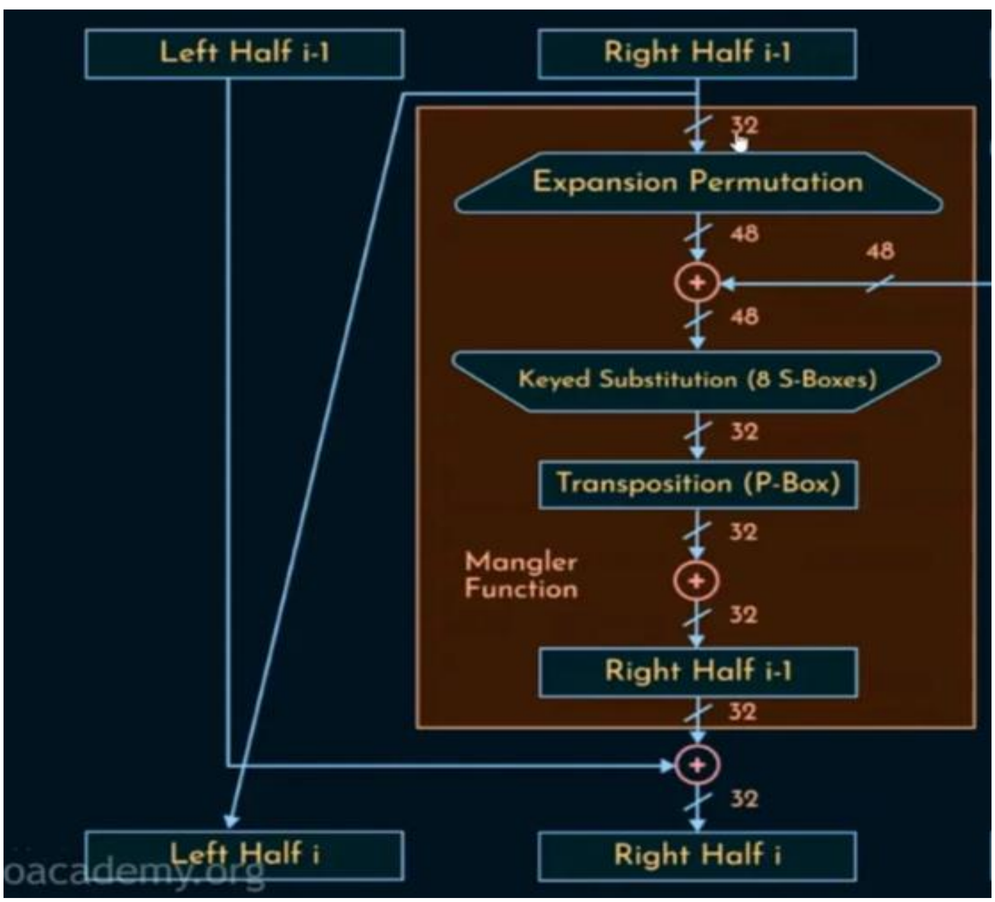


Steps of Single Round Operation:

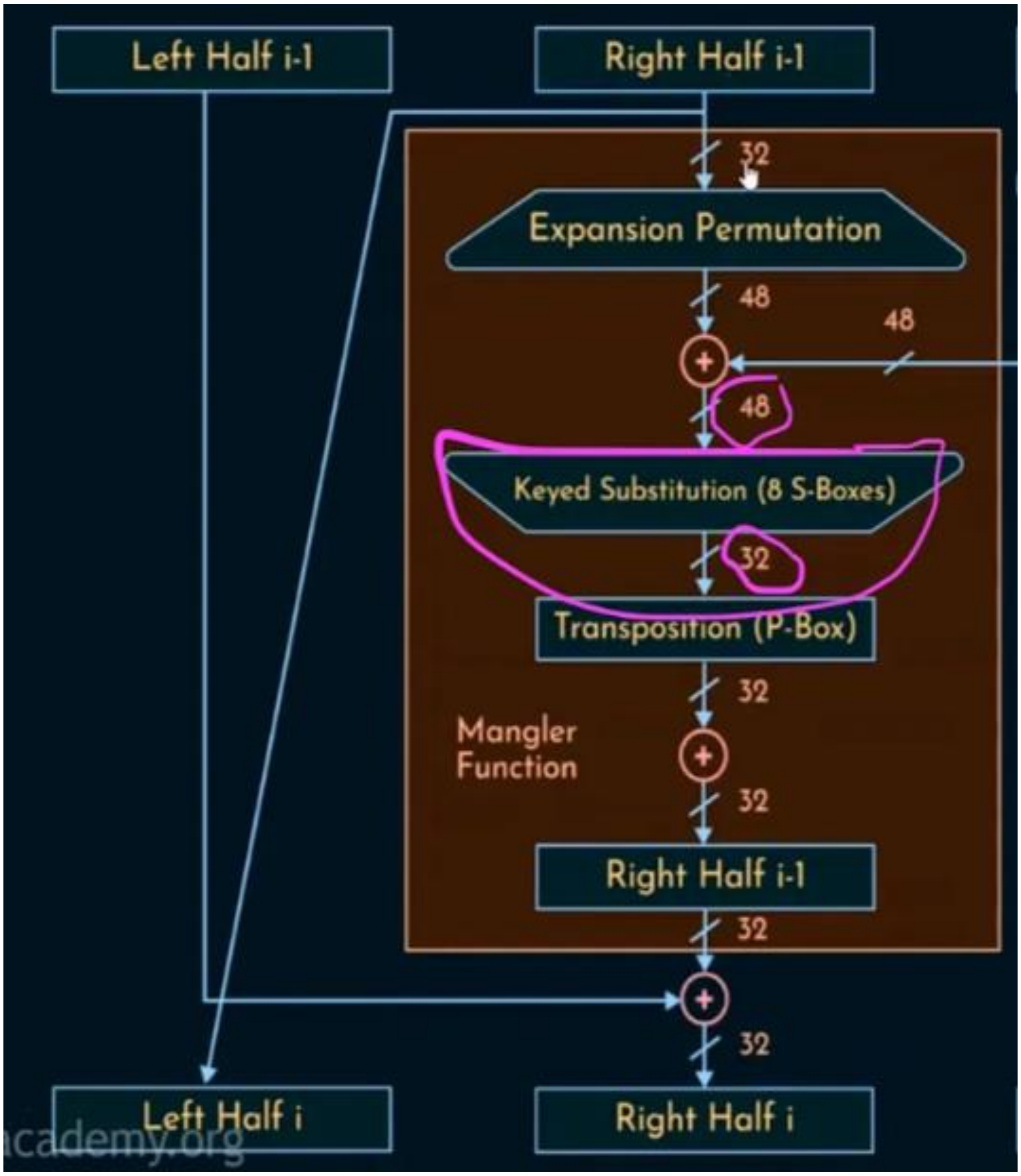
1. In the above figure, Left Hand i -1 takes 32-bit Right Hand Side takes Half i-1 32-bit, and the Right-Hand Half Side 32-bit is given to the expansion permutation function.

2. The expansion Permutation Function expands the 32-bit into 48-bits because we are getting the 48-bit key and our original input size was 64-bit.

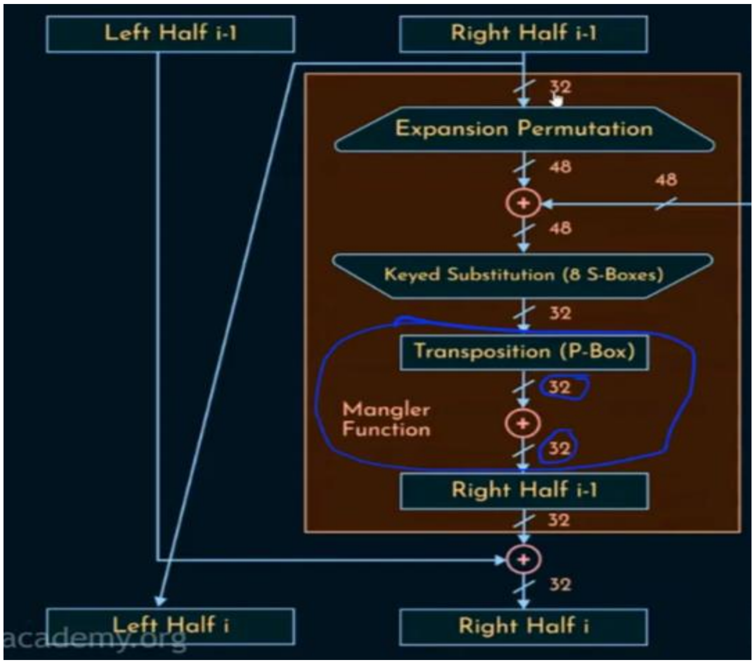
3. The Right Side 32-bit has to be converted into 48-bit so that it can be XORED with the key 48-bit.



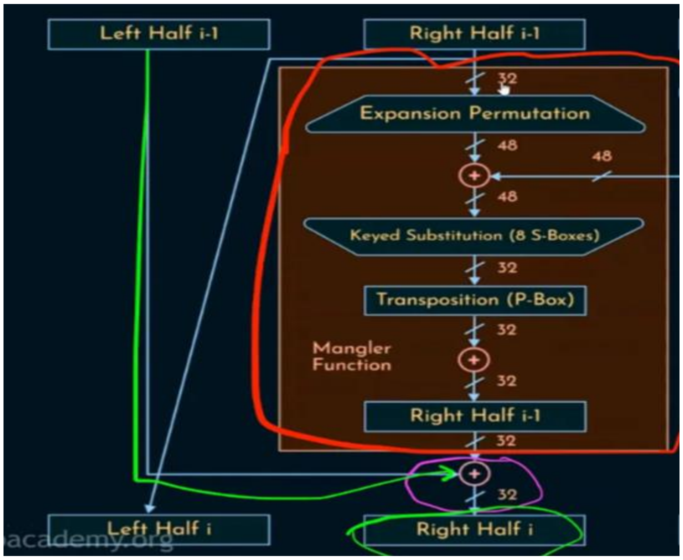
4. Again 8 S-Box reduces the input size of 48-bit to 32-bit and the output of the S-Box is given to the P-Box (Transposition).



5. P-Box changes the position of the bit and again gives to the right half i - 1 which performs the XORed Operation.



6. Now the right half i-1 32-bit is XORed with the left-hand i -1 and the output is the 32-bit Whole Right Hand Half part. That means the figure shows the whole red bordered part output XORed with the left hand i -1 32-bit input gives the Right Half Output i.

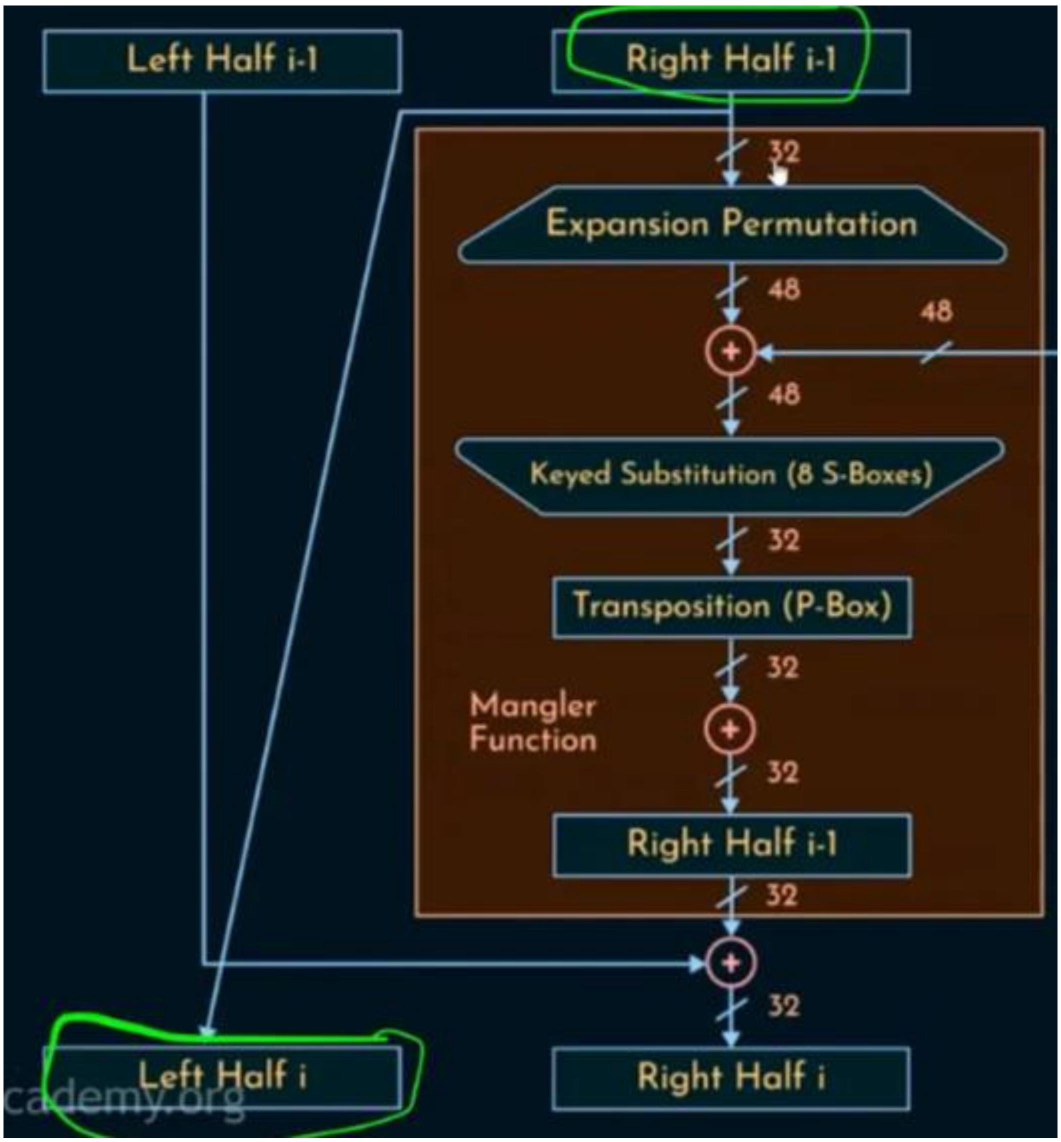


Note: Now we get the Right Half i and the Left Hand i, these two 32-bit will be the input for another second-round operation up to 16-round operations. That means if we observe in the above figure before the expansion permutation, S-box substitution, transposition, and right half i -1 right half i -1 is the output for left half i which will be the input for another round operation.

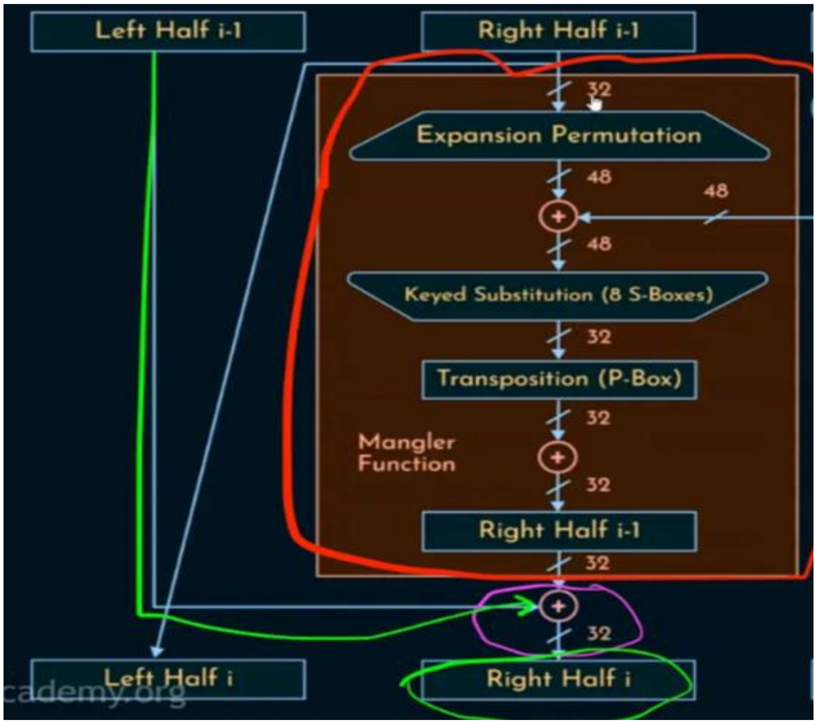
The Whole Steps of a Single Round Operation will be performed 16 times because DES uses 16 Round Operations.

If we try to outline the equation of the left and right-hand sides. See the green bordered part in the figure.

Li = R i-1

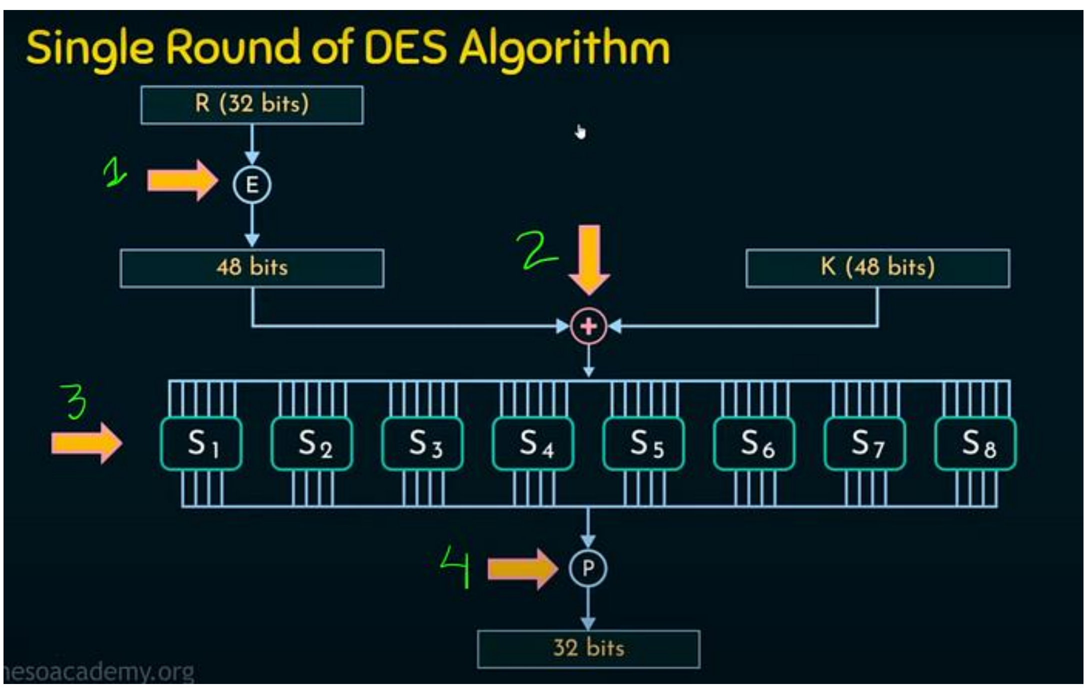


Then if we again look at the red bordered part as Mangler Function denoted by F we get the equation as Ri = Li-1 ⊕ F (R_i-1, Ki)



Mangler Function

The Mangler Function F is the **F (Ri-1, Ki).**



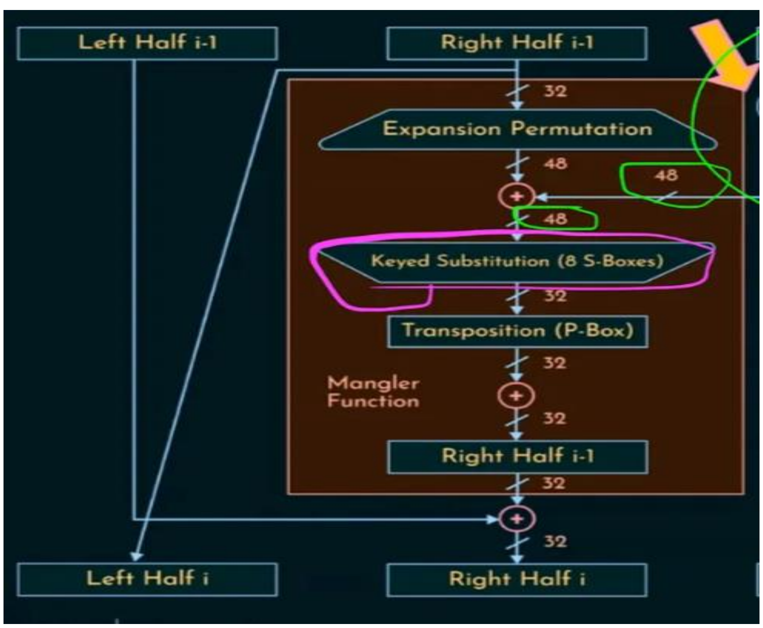
How Does Mangler Function / F Function Work?

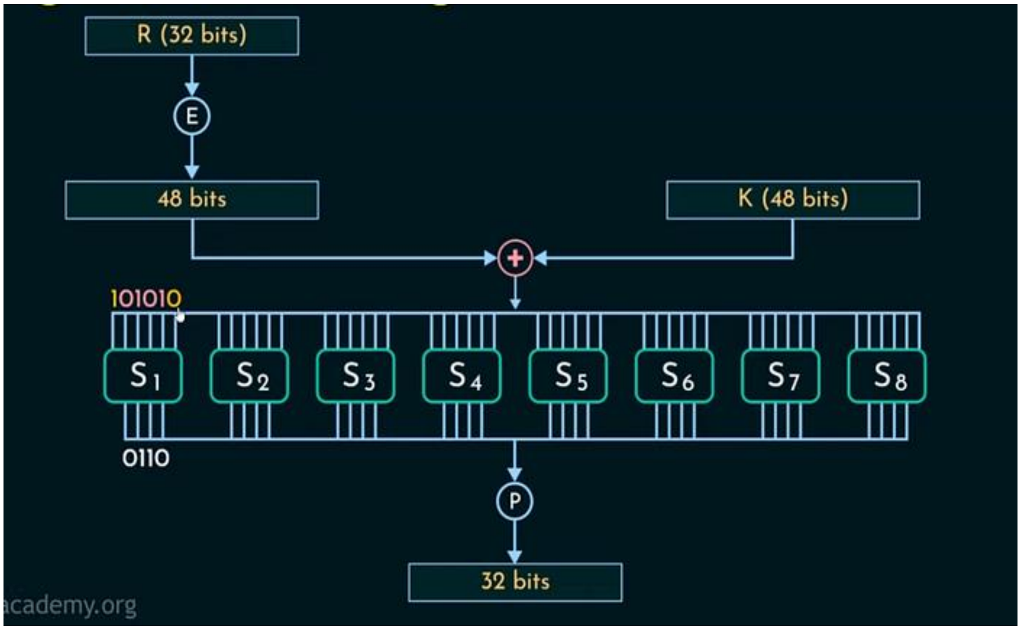
- In the above figure green colored is step 1 is 32 Bits we need to Convert it into again 48 Bits.
- For converting 32 Bits to 48 Bits, we have to use below table which is known as the Expansion Permutation table.

The Expansion Permutation

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

- By arranging the right-hand sides 32 Bits According to this E-box we get 48 Bits.
- Now the R (32 Bit) operation is completed and we have the output of 48 bits from the R (32 Bits).
- We have to perform the output of R (32 Bits) XOR with the K (48 bits) the 48 Bits.
- After the XOR again we have to send them to the S-Box a purple-colored border part box. S-box takes 8 blocks of input 6 bits and output 8 blocks of 4 bits which makes it 32 bits.





How 6-bit Input in the S-Box Block is converted into 4-Bit Block?

Box S_1

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	6	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

For example, S_1(101010) = 6 = 0110.

- The figure shows all the 6-bit of 8 blocks. and this is for only S1. S1 is taking 101010 input and giving output as 0110.
- The first and last bit from the S1 = 101010 is 1 and 0 that is (10 which is a row in the table and the remaining 0101 is a column in the table).
- The value corresponding to row 10 and column 0101 is 6 (0101).
- In this way, we perform S-Box for all 8 S-blocks. The output 32-bit we will obtain from the S-box is again permutated.

Permutation Function Applied to the 32-bit Output from S-box

- The 32-bit output positions are only changed again and shuffled using the permutation function (P) which table is given below.

The 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

Phase 4

- Finally, after 16 rounds we get L_{16} (Left Half after 16 Rounds) and R_{16} (Right Half after 16 Rounds):

  L_{16} (Left Half after 16 Rounds): 0100 0011 0100 0010 0011 0010 0011 0100

  R_{16} (Right Half after 16 Rounds): 0000 1010 0100 1100 1101 1001 1001 0101
- In DES, before the final permutation, $\mathbf{L_{16}}$ and $\mathbf{R_{16}}$ are swapped:
- Concatenated as $\mathbf{R_{16}L_{16}}$: 0000 1010 0100 1100 1101 1001 1001 0101 0100 0011 0100 0010 0011 0010 0011 0100
- Applying the inverse of the initial permutation (IP^{-1}) table on $\mathbf{R_{16}L_{16}}$: we get 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

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

Now convert each 8-bit binary group to hexadecimal:

- 10000101 → 85
- 11101000 → E8
- 00010011 → 13
- 01010100 → 54
- 00001111 → 0F
- 00001010 → 0A
- 10110100 → B4
- 00000101 → 05

Cipher text in hexadecimal format is 85E813540F0AB405.

Like the we encrypt every block of message.

The truth

- DES is in secure as it uses a 56-bit key, which is now considered too short to withstand brute-force attacks.
- In **January 1999**, **distributed.net** and the **Electronic Frontier Foundation (EFF)** broke a DES key in just **22 hours and 15 minutes**, proving its vulnerability.
- DES has been **superseded by the Advanced Encryption Standard (AES)**, which offers stronger security.
- DES has been officially **withdrawn as a standard by the National Institute of Standards and Technology (NIST)**.