Data Encryption Standard (DES)

The Data Encryption Standard (DES) is **a symmetric-key block cipher** published by the National Institute of Standards and Technology (NIST). It has a 56-bit key length which has played a significant role in data security**.** Data encryption standard (DES) has been found vulnerable to very powerful attacks therefore, the popularity of DES has been found slightly on the decline. DES is a block cipher and encrypts data in blocks of size of **64 bits** each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is **56 bits**.

The basic idea is shown below:

We have mentioned that DES uses a 56-bit key. Actually, The initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.

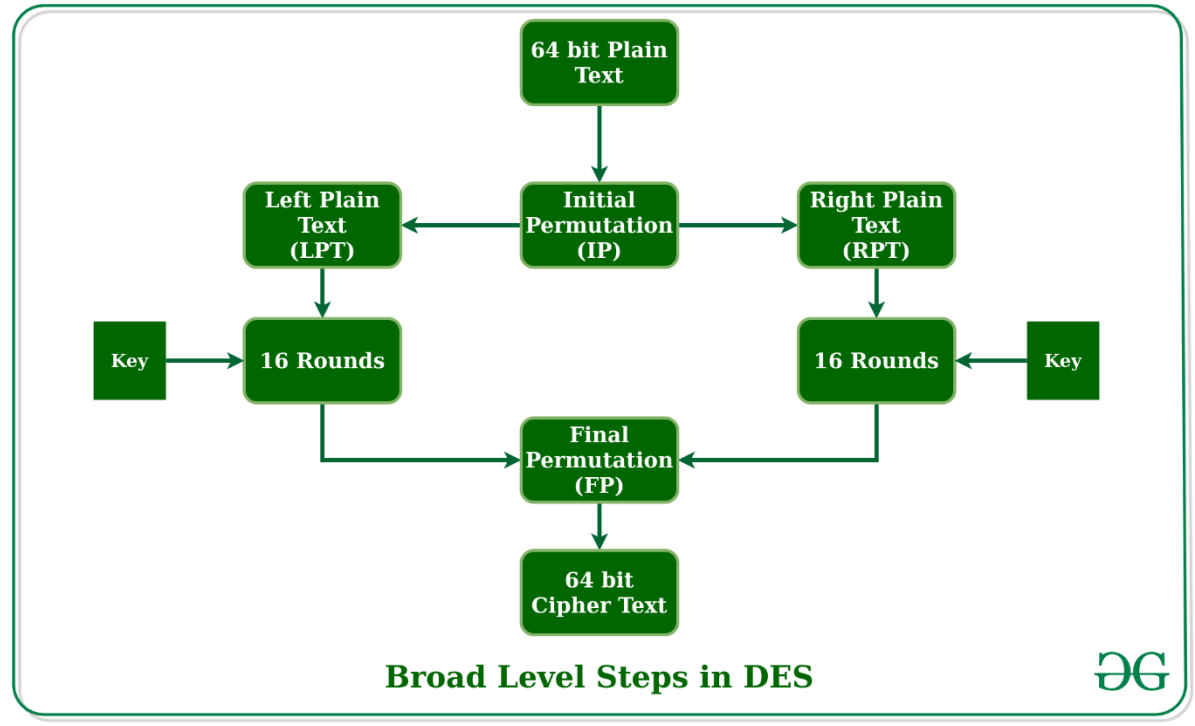
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Figure - discording of every 8th bit of original key

Thus, the discarding of every 8th bit of the key produces a **56-bit key** from the original **64-bit key**.

DES is based on the two fundamental attributes of cryptography: substitution (also called confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

- In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
- The initial permutation is performed on plain text.
- Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).
- Now each LPT and RPT go through 16 rounds of the encryption process.
- In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64-bit ciphertext.



Initial Permutation (IP)

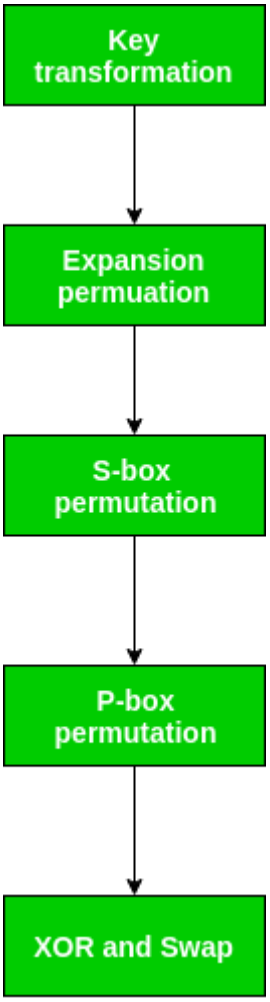
As we have noted, the initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in the figure. For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block, and so on.

This is nothing but jugglery of bit positions of the original plain text block. the same rule applies to all the other bit positions shown in the figure.

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	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure - Initial permutation table

As we have noted after IP is done, the resulting 64-bit permuted text block is divided into two half blocks. Each half-block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad-level steps outlined in the figure.



tep 1: Key transformation

We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called key transformation. For this, the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

For example: if the round numbers 1, 2, 9, or 16 the shift is done by only one position for other rounds, the circular shift is done by two positions. The number of key bits shifted per round is shown in the figure.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
#key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Figure - number of key bits shifted per round

After an appropriate shift, 48 of the 56 bits are selected. From the 48 we might obtain 64 or 56 bits based on requirement which helps us to recognize that this model is very versatile and can

handle any range of requirements needed or provided. for selecting 48 of the 56 bits the table is shown in the figure given below. For instance, after the shift, bit number 14 moves to the first position, bit number 17 moves to the second position, and so on. If we observe the table , we will realize that it contains only 48-bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as a selection of a 48-bit subset of the original 56-bit key it is called Compression Permutation.

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

Figure - compression permutation

Because of this compression permutation technique, a different subset of key bits is used in each round. That makes DES not easy to crack.

Step 2: Expansion Permutation

Recall that after the initial permutation, we had two 32-bit plain text areas called Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called expansion permutation. This happens as the 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4-bit block of the previous step is then expanded to a corresponding 6-bit block, i.e., per 4-bit block, 2 more bits are added.

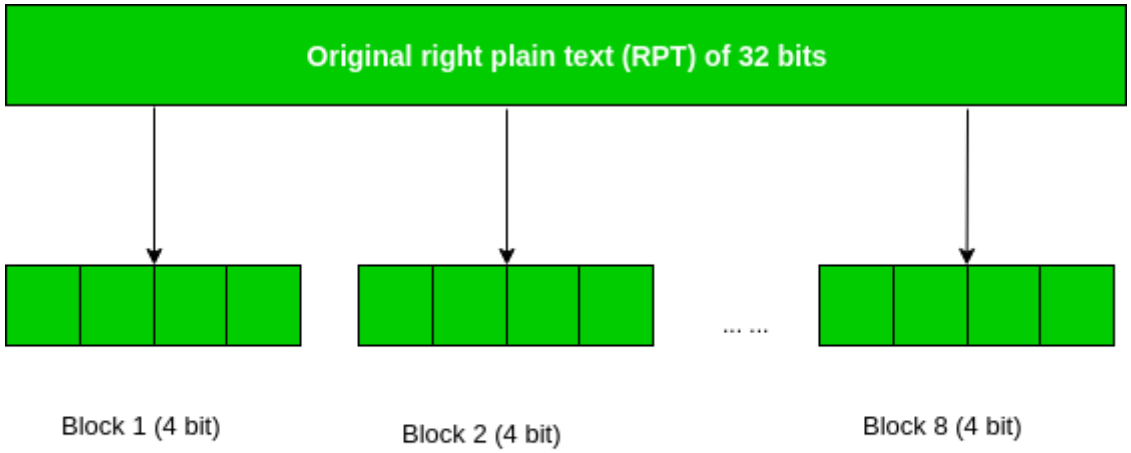


Figure - division of 32 bit RPT into 8 bit blocks

This process results in expansion as well as a permutation of the input bit while creating output. The key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the **32-bit RPT** to **48-bits**. Now the 48-bit key is XOR with 48-bit RPT and the resulting output is given to the next step, which is the **S-Box substitution**.

Program 5: Write a C program to implement DES algorithm.

```
#include<stdio.h>
#include<string.h>

int s_matrix0[4][4] = { {1,0,3,2},
                        {3,2,1,0},
                        {0,2,1,3},
                        {3,1,3,2} };

int s_matrix1[4][4] = { {0,1,2,3},
                        {2,0,1,3},
                        {3,0,1,0},
                        {2,1,0,3} };

int s0=0, s1=0;
int row=0, col=0;
int s0_binary[2], s1_binary[2];
int result[2];

int to_digit(int a, int b)
{
    int output;
    if(a==1 && b==1)
        output = 3;

    if(a==0 && b==1)
        output = 1;

    if(a==1 && b==0)
        output = 2;
```

```
    if(a==0 && b==0)
        output = 0;

    return output;
}

void to_binary(int num)
{
    int x;

    if(num == 3)
    {
        for(x=0; x<2; x++)
            result[x] = 1;
    }
    else if(num == 1)
    {
        result[0] = 0;
        result[1] = 1;
    }
    else if(num == 2)
    {
        result[0] = 1;
        result[1] = 0;
    }
    else if(num == 0)
    {
        for(x=0; x<2; x++)
            result[x] = 0;
    }
}

void main()
{
    int i,j,index;
```

```
    int k1[8], k2[8];
    int afterp10[11], ls1[10], ls2[10], afterip[8], afterep[8],
afterp4[4];
    int aftersboxesone[4], aftersboxestwo[4];
    int leftafterip[4], rightafterip[4];
    int leftafterep[4], rightafterep[4];
    int leftafterone[4], rightafterone[4];
    int afteripinverse[8];
    int afterone[8], aftertwo[8];

    int p10[10] = {3,5,2,7,4,10,1,9,8,6};
    int p8[8] = {6,3,7,4,8,5,10,9};

    int ip[8] = {2,6,3,1,4,8,5,7};
    int ep[8] = {4,1,2,3,2,3,4,1};
    int p4[4] = {2,4,3,1};
    int ipinverse[8] = {4,1,3,5,7,2,8,6};

    int key[11] = {1,0,1,0,0,0,0,0,1,0};
    int plain[8] = {0,1,1,0,1,1,0,1};

    printf("S-DES Key Generation and Encryption.\n");

    printf("\n----------KEY GENERATION----------\n");
    printf("\nEntered the 10-bit key is: ");
    for(i=0; i<10; i++)
        printf("%d ",key[i]);

    printf("\np10 permutation is defined as: ");
    for(i=0; i<=9; i++)
        printf("%d ",p10[i]);

    for(i=0; i<=9; i++)
    {
        index = p10[i];
        afterp10[i] = key[index - 1];
    }
```

```
afterp10[i] = '\0';

printf("\n\nAfter p10 = ");
for(i=0; i<=9; i++)
    printf("%d ",afterp10[i]);

for(i=0; i<5; i++)
{
    if(i == 4)
        ls1[i]=afterp10[0];
    else
        ls1[i]=afterp10[i+1];
}

for(i=5; i<10; i++)
{
    if(i == 9)
        ls1[i]=afterp10[5];
    else
        ls1[i]=afterp10[i+1];
}

printf("\n\nAfter LS-1 = ");
for(i=0; i<10; i++)
    printf("%d ", ls1[i]);

printf("\np8 permutation is defined as: ");
for(i=0;i<8;i++)
    printf("%d ",p8[i]);

index=0;
for(i=0; i<=9; i++)
{
    index = p8[i];
    k1[i] = ls1[index - 1];
}
```

```
printf("\n\n---->Key-1 is: ");
for(i=0;i<8;i++)
    printf("%d ",k1[i]);

for(i=0; i<3; i++)
{
    ls2[i]=ls1[i+2];
}
ls2[3]=ls1[0];
ls2[4]=ls1[1];

for(i=5; i<8; i++)
{
    ls2[i]=ls1[i+2];
}
ls2[8]=ls1[5];
ls2[9]=ls1[6];

printf("\n\nAfter LS-2 = ");
for(i=0; i<10; i++)
    printf("%d ", ls2[i]);

printf("\np8 permutation is defined as: ");
for(i=0;i<8;i++)
    printf("%d ",p8[i]);

index=0;
for(i=0; i<=9; i++)
{
    index = p8[i];
    k2[i] = ls2[index - 1];
}

printf("\n\n---->Key-2 is: ");
for(i=0;i<8;i++)
    printf("%d ",k2[i]);
```

```
printf("\n\n----------S-DES ENCRYPTION---------\n");
printf("\n---->Entered the 8-bit plaintext is: ");
for(i=0; i<8; i++)
    printf("%d ",plain[i]);


printf("\nInitial permutation is defined as: ");
for(i=0; i<8; i++)
    printf("%d ",ip[i]);

for(i=0; i<8; i++)
{
    index = ip[i];
    afterip[i] = plain[index - 1];
}
afterip[i] = '\0';

printf("\nAfter ip = ");
for(i=0; i<8; i++)
    printf("%d ", afterip[i]);

printf("\n\nExpand permutation is defined as: ");
for(i=0; i<8; i++)
    printf("%d ",ep[i]);

for(j=0; j<4; j++)
    leftafterip[j] = afterip[j];

for(j=0; j<4; j++)
    rightafterip[j] = afterip[j+4];

for(i=0; i<4; i++)
{
    index = ep[i];
    afterep[i] = rightafterip[index - 1];
}
```

```
for(i=4; i<8; i++)
{
    index = ep[i];
    afterep[i] = rightafterip[index - 1];
}
afterep[i] = '\0';

printf("\nAfter ep = ");
for(i=0; i<8; i++)
    printf("%d ", afterep[i]);

for(i=0; i<8; i++)
    k1[i] = k1[i] ^ afterep[i];

printf("\n\nAfter XOR operation with 1st Key= ");
for(i=0; i<8; i++)
    printf("%d ", k1[i]);

row = to_digit(k1[0],k1[3]);
col = to_digit(k1[1],k1[2]);
s0 = s_matrix0[row][col];

to_binary(s0);
for(j=0; j<2; j++)
    s0_binary[j] = result[j];

row = to_digit(k1[4],k1[7]);
col = to_digit(k1[5],k1[6]);

s1 = s_matrix1[row][col];

to_binary(s1);
for(j=0; j<2; j++)
    s1_binary[j] = result[j];

for(j=0; j<2; j++)
    aftersboxesone[j] = s0_binary[j];
```

```
    for(i=0,j=2; i<2,j<4; i++,j++)
        aftersboxesone[j] = s1_binary[i];

    printf("\n\nAfter first S-Boxes= ");
    for(i=0; i<4; i++)
        printf("%d ", aftersboxesone[i]);

    printf("\n\nP4 is defined as: ");
    for(i=0; i<4; i++)
        printf("%d ",p4[i]);

    for(i=0; i<4; i++)
    {
        index = p4[i];
        afterp4[i] = aftersboxesone[index - 1];
    }
    afterp4[i] = '\0';

    printf("\nAfter P4= ");
    for(i=0; i<4; i++)
        printf("%d ", afterp4[i]);

    for(i=0; i<4; i++)
        afterp4[i] = afterp4[i] ^ leftafterip[i];

    printf("\n\nAfter XOR operation with left nibble of after ip=
");
    for(i=0; i<4; i++)
        printf("%d ", afterp4[i]);

    for(i=0; i<4; i++)
        afterone[i] = rightafterip[i];

    for(i=0,j=4; i<4,j<8; i++,j++)
        afterone[j] = afterp4[i];
```

```
afterone[j] = '\0';

printf("\nAfter first part= ");
for(i=0; i<8; i++)
    printf("%d ", afterone[i]);

for(j=0; j<4; j++)
    leftafterone[j] = afterone[j];

for(j=0; j<4; j++)
    rightafterone[j] = afterone[j+4];

printf("\n\nExpand permutation is defined as: ");
for(i=0; i<8; i++)
    printf("%d ",ep[i]);

for(i=0; i<4; i++)
{
    index = ep[i];
    afterep[i] = rightafterone[index - 1];
}

for(i=4; i<8; i++)
{
    index = ep[i];
    afterep[i] = rightafterone[index - 1];
}
afterep[i] = '\0';

printf("\nAfter second ep = ");
for(i=0; i<8; i++)
    printf("%d ", afterep[i]);

for(i=0; i<8; i++)
    k2[i] = k2[i] ^ afterep[i];

printf("\n\nAfter XOR operation with 2nd Key= ");
```

```
for(i=0; i<8; i++)
    printf("%d ", k2[i]);

row = to_digit(k2[0],k2[3]);
col = to_digit(k2[1],k2[2]);

s0 = s_matrix0[row][col];
to_binary(s0);
for(j=0; j<2; j++)
    s0_binary[j] = result[j];

row = to_digit(k2[4],k2[7]);
col = to_digit(k2[5],k2[6]);

s1 = s_matrix1[row][col];
to_binary(s1);
for(j=0; j<2; j++)
    s1_binary[j] = result[j];

for(j=0; j<2; j++)
    aftersboxestwo[j] = s0_binary[j];

for(i=0,j=2; i<2,j<4; i++,j++)
    aftersboxestwo[j] = s1_binary[i];

printf("\n\nAfter second S-Boxes= ");
for(i=0; i<4; i++)
    printf("%d ", aftersboxestwo[i]);

printf("\n\nP4 is defined as: ");
for(i=0; i<4; i++)
    printf("%d ",p4[i]);

for(i=0; i<4; i++)
{
    index = p4[i];
    afterp4[i] = aftersboxestwo[index - 1];
```

```
    }
    afterp4[i] = '\0';

    printf("\nAfter P4= ");
    for(i=0; i<4; i++)
        printf("%d ", afterp4[i]);

    for(i=0; i<4; i++)
        afterp4[i] = afterp4[i] ^ leftafterone[i];

    printf("\n\nAfter XOR operation with left nibble of after first
part= ");
    for(i=0; i<4; i++)
        printf("%d ", afterp4[i]);

    for(i=0; i<4; i++)
        aftertwo[i] = afterp4[i];

    for(i=0,j=4; i<4,j<8; i++,j++)
        aftertwo[j] = rightafterone[i];

    aftertwo[j] = '\0';

    printf("\n\nAfter second part= ");
    for(i=0; i<8; i++)
        printf("%d ", aftertwo[i]);


    printf("\n\nInverse Initial permutation is defined as: ");
    for(i=0; i<8; i++)
        printf("%d ",ipinverse[i]);

    for(i=0; i<8; i++)
    {
        index = ipinverse[i];
        afteripinverse[i] = aftertwo[index - 1];
    }
```

```
    afteripinverse[j] = '\0';

    printf("\n\n--->8-bit Ciphertext will be= ");
    for(i=0; i<8; i++)
        printf("%d ", afteripinverse[i]);
}
```

Output:

```
S-DES Key Generation and Encryption.

----------KEY GENERATION----------

Entered the 10-bit key is: 1 0 1 0 0 0 0 0 1 0
p10 permutation is defined as: 3 5 2 7 4 10 1 9 8 6

After p10 = 1 0 0 0 0 0 1 1 0 0

After LS-1 = 0 0 0 0 1 1 1 0 0 0
p8 permutation is defined as: 6 3 7 4 8 5 10 9

---->Key-1 is: 1 0 1 0 0 1 0 0

After LS-2 = 0 0 1 0 0 0 0 0 1 1
p8 permutation is defined as: 6 3 7 4 8 5 10 9

---->Key-2 is: 0 1 0 0 0 0 1 1

----------S-DES ENCRYPTION---------

---->Entered the 8-bit plaintext is: 0 1 1 0 1 1 0 1
Initial permutation is defined as: 2 6 3 1 4 8 5 7
After ip = 1 1 1 0 0 1 1 0

Expand permutation is defined as: 4 1 2 3 2 3 4 1
After ep = 0 0 1 1 1 1 0 0

After XOR operation with 1st Key= 1 0 0 1 1 0 0 0

After first S-Boxes= 1 1 1 1

P4 is defined as: 2 4 3 1
After P4= 1 1 1 1

After XOR operation with left nibble of after ip= 0 0 0 1
After first part= 0 1 1 0 0 0 0 1

Expand permutation is defined as: 4 1 2 3 2 3 4 1
After second ep = 1 0 0 0 0 0 1 0

After XOR operation with 2nd Key= 1 1 0 0 0 0 0 1

After second S-Boxes= 0 1 1 0

P4 is defined as: 2 4 3 1
After P4= 1 0 1 0

After XOR operation with left nibble of after first part= 1 1 0 0

After second part= 1 1 0 0 0 0 0 1

Inverse Initial permutation is defined as: 4 1 3 5 7 2 8 6

--->8-bit Ciphertext will be= 0 1 0 0 0 1 1 0
Process returned 2 (0x2)   execution time : 0.096 s
Press any key to continue.
```