**4 Program: IMPLEMENTATION OF DES**

**AIM:**

To write a C program to implement Data Encryption Standard (DES) using C

Language.

**DESCRIPTION:**

DES is a symmetric encryption system that uses 64-bit blocks, 8 bits of which are

used for parity checks. The key therefore has a "useful" length of 56 bits, which means that

only 56 bits are actually used in the algorithm. The algorithm involves carrying out

combinations, substitutions and permutations between the text to be encrypted and the key,

while making sure the operations can be performed in both directions. The key is ciphered on

64 bits and made of 16 blocks of 4 bits, generally denoted k1 to k16. Given that "only" 56 bits

are actually used for encrypting, there can be 256 different keys.

**The main parts of the algorithm are as follows:**

\_ Fractioning of the text into 64-bit blocks

\_ Initial permutation of blocks

\_ Breakdown of the blocks into two parts: left and right, named L and R

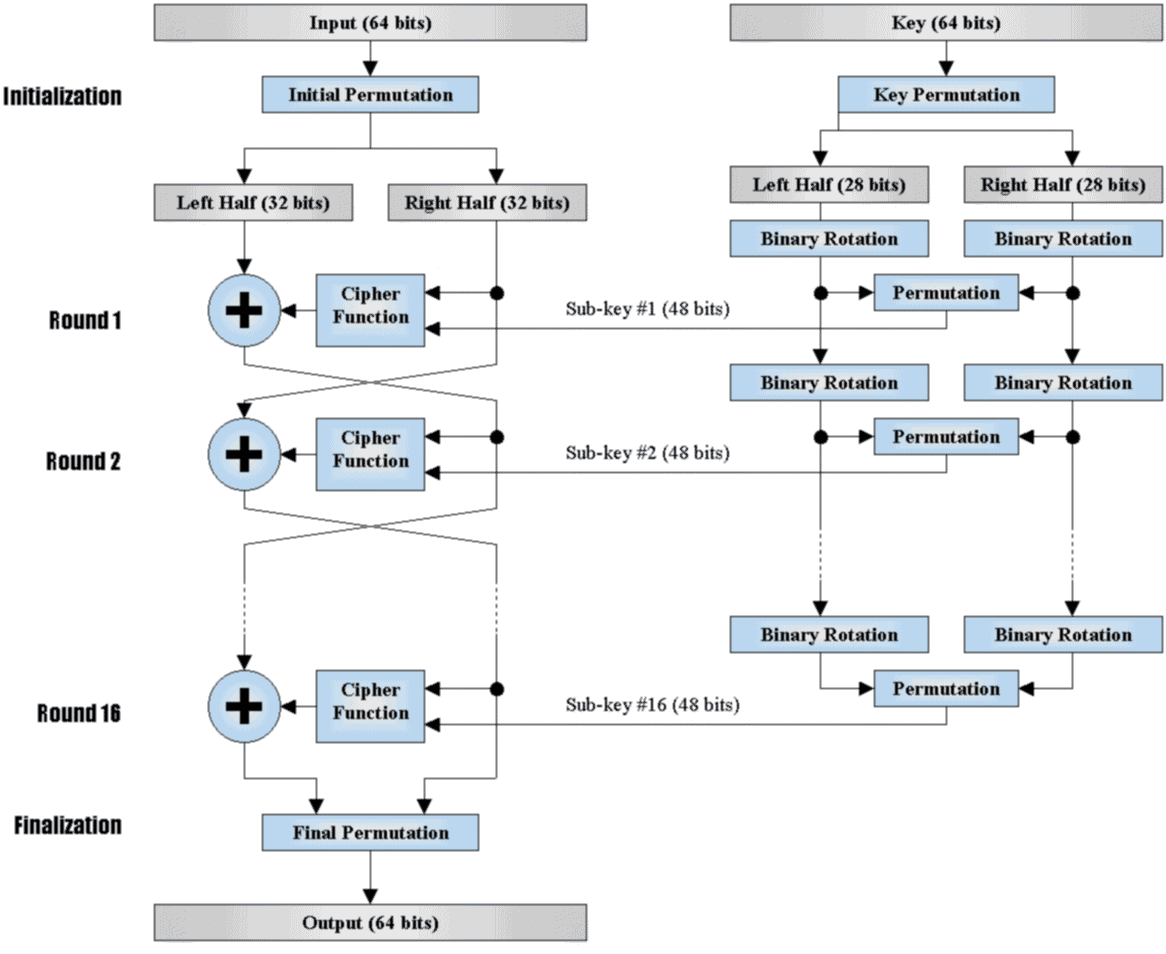
\_ Permutation and substitution steps repeated 16 times

\_ Re-joining of the left and right parts then inverse initial permutation

**DES:**

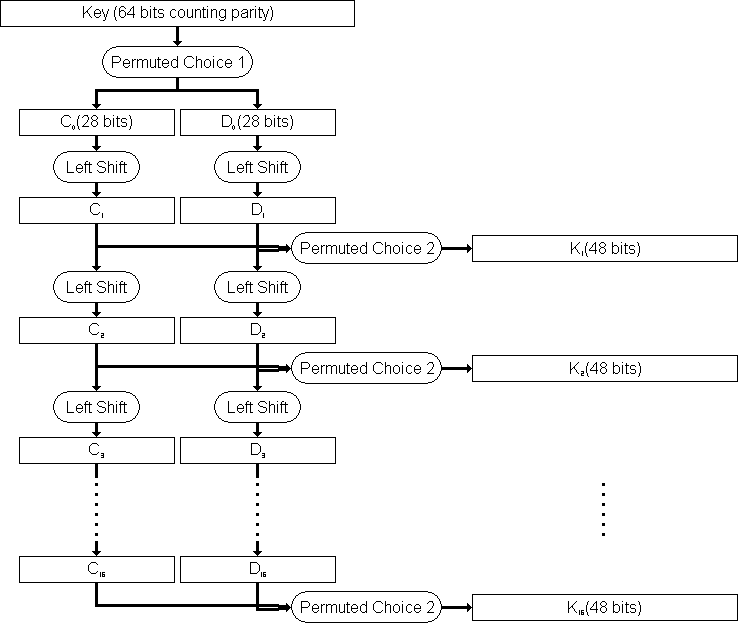
DES adopted in 1977 by (NIST) National Institute of Slandered and Technology).

With DES, data are encrypted in 64 bit blocks, with a key length 56 bits and with output ciphertext 64 bits. DES has the exact structure of Feistel Cipher, but without Initial Permutation (IP) and Inverse Initial Permutation *IP*−1.

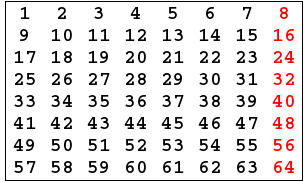
[](https://www.cybrary.it/wp-content/uploads/2016/01/des-enc.gif)

**Key Generator Algorithm:**

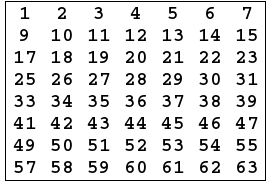
The key generator passes through many steps to produce sub keys.

[](https://www.cybrary.it/wp-content/uploads/2016/01/des-key.gif)

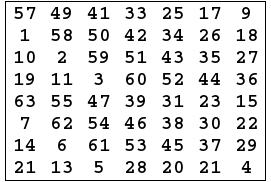
**1-** The key generator algorithm takes 64 bits key as input. The input key number table from 1 to 64 is as follows:

[](https://www.cybrary.it/wp-content/uploads/2016/01/1.png)

**2-** Every eighth bit is ignored and produces 56 bits.

[](https://www.cybrary.it/wp-content/uploads/2016/01/2.png)

**3-** 56 bits pass through a permutation Choice one (PC-1) and displays as follows:

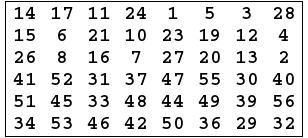
[](https://www.cybrary.it/wp-content/uploads/2016/01/3.png)

**4-** The output is separated into two 28 bits C and D. The he first 28 bits are called  *C*0 (left part) and the last 28 bits are called *D*0.

**5-** At each round, a circular left shift is preformed on  *Ci*−1 and  *Di*−1 by 1 or 2 bits. See the table below:

[https://www.cybrary.it/wp-content/uploads/2016/01/4.png](https://www.cybrary.it/wp-content/uploads/2016/01/4.png)

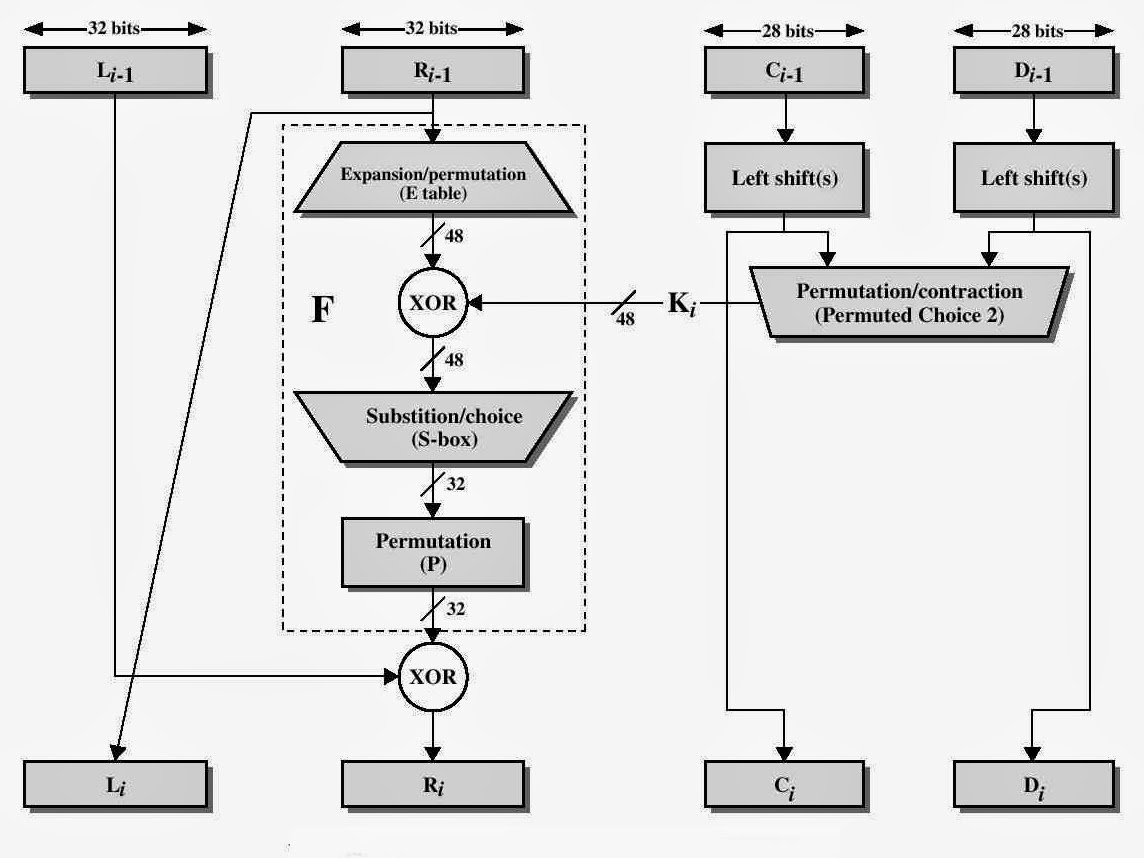
**6-** Then *Ci*−1 and  *Di*−1 in each round, passes through permutation choice two (PC-2) to produce 48 bits.

[](https://www.cybrary.it/wp-content/uploads/2016/01/5.png)

**7-** The permutation Choice Two output in each round is uses as input to the encryption algorithm.

**Encryption Algorithm:**

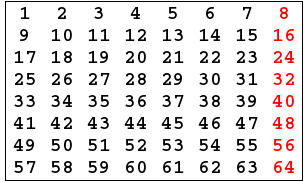
There are two inputs to the encryption algorithm: 1- Plaintext 64 bits 2- Encryption key 48 bits.  
The encryption algorithm also passes through many steps to produce a cipher text. See the figure below:

[](https://www.cybrary.it/wp-content/uploads/2016/01/des-enc3.jpg)

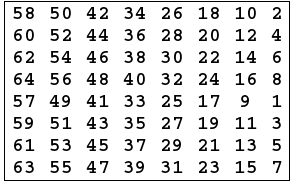
**1-** The plaintext block 64 bits pass through an initial permutation (IP) that the rearranged bits and produces the permuted input.

**Initial Permutation:**

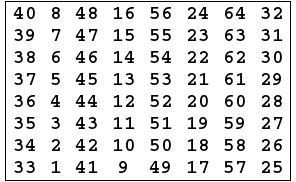
**A-** Initial Permutation takes the plaintext as input. The table consists of 64 bits numbered from 1 to 64:

[](https://www.cybrary.it/wp-content/uploads/2016/01/11.png)

**B-** Then the initial permutation will be permuted input as 64 bits:

[](https://www.cybrary.it/wp-content/uploads/2016/01/6.png)

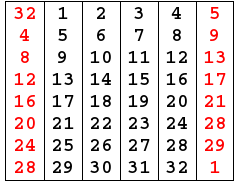
**C-** The Inverse Initial Permutation is:

[](https://www.cybrary.it/wp-content/uploads/2016/01/7.png)

**2-** The permuted input block split into two halves each is 32 bits. The first 32 bits are called L and the last 32 bits are called R.

*Now, The F function will start the rest of all the steps.*

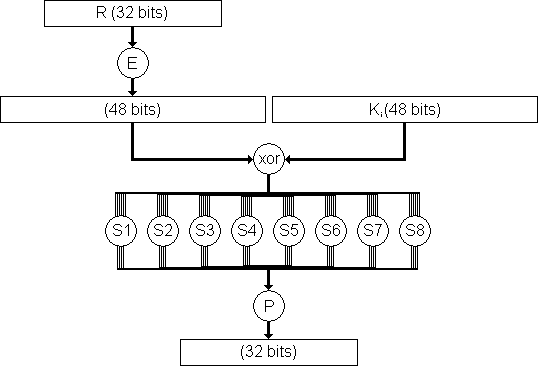
**3-** Expand R 32 bits to 48 bits to fit the subkey by preforming the Expansion permutation (E):

[](https://www.cybrary.it/wp-content/uploads/2016/01/8.png)

**4-** Perform Exclusive-OR between the subkey and Expansion Permutation (E) on R.  
E(*Ri*-1)⊕ *Ki*.

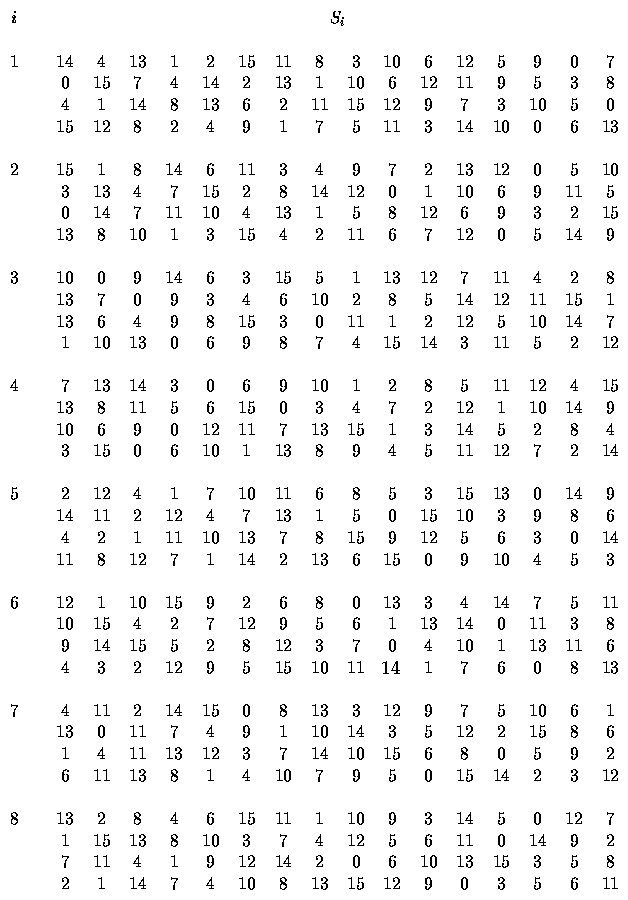
**5-** The result of E(*Ri*-1)⊕ *Ki* pass through a substitution function and produce 32 bits output.

**Substitution Function:**

[](https://www.cybrary.it/wp-content/uploads/2016/01/des-sub.gif)

Substitution Function is rolled by S-Box. S-Box consists of 8 boxes, each of which accepts 6 bits as input and produces 4 bits output:

**A-** Break the result of E(*Ri*-1)⊕ *Ki* into 8 blocks, each containing 6 bits. These blocks are numbered from 1 to 8.  
**B-** Each block will perform a substitution with S-Box with the same number:

[](https://www.cybrary.it/wp-content/uploads/2016/01/des-s-box2.gif)

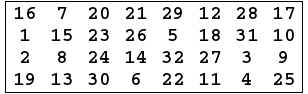
**A-** The first and the last bits of each block together as 2 bit value indicate the number of rows in the same number S-Box.  
**B-** The middle four bits of each block together as-bit value, indicating the number of columns in the same number S-Box.  
**C-** The decimal value, which is selected by the row and the column convert to-bit value in all S-Boxes.

**For Example:**

Suppose the first 6 bits of the result of E(*Ri*-1)⊕ *Ki* = 010101.  
So, the input to *S*1 = 010101.  
The row value = 0 1 = 1 (decimal).  
The column value = 1010 = 10 (decimal).  
The decimal value will be 12 = 1100 (4-bit value).

**D-** Combine results of each S-Box together 32 bits.

**6-** The result of the substitution operation (output of S-Boxes) passes through a Permutation Function (P).

[](https://www.cybrary.it/wp-content/uploads/2016/01/9.png)

*At this point, the function F is finished.*

**7- Perform Exclusive-OR between the output of the Permutation Function(P) and *Li*−1. Then, put the result in *Ri* , and put *Ri*−1 in*Li*.**

**The overall formulas for DES Encryption Algorithm:**

*Li* = *Ri*−1.  
*Ri* = *Li*−1 ⊕ F(*Ri*−1,*Ki*).

**8-** Perform a 32-bit swap on the result of the final round. Then, perform Inverse Initial Permutation (*IP*−1) on the swapped data to produces the ciphertext 64 bits.

**Decryption Algorithm:**

The inputs to decryption algorithm are ciphertext and subkey *Ki* but in reverse order, start with *Kn* then *K*(*n*−1) and so on until *K*1 in the last round.

**ALGORITHM:**

**STEP-1:** Read the 64-bit plain text.

**STEP-2:** Split it into two 32-bit blocks and store it in two different arrays.

**STEP-3:** Perform XOR operation between these two arrays.

**STEP-4:** The output obtained is stored as the second 32-bit sequence and the original

second 32-bit sequence forms the first part.

**STEP-5:** Thus the encrypted 64-bit cipher text is obtained in this way. Repeat the same

process for the remaining plain text characters.