**Lab 2**

**Lab 2.1: Write a program to implement the DES key generation process to generate subkeys. Also, show the subkeys generated at each round.**

**Algorithm:**

DES key generation process to generate subkeys:

1. Input:
   1. Accept a 64-bit key as input.
2. Initial permutation (PC1):
   1. Apply the PC1 (Permuted Choice 1) table to permute the 64-bit key, rearranging its bits to create a 56-bit key.
3. Key splitting:
   1. Divide the 56-bit key into two halves of 28 bits each, C0 and D0.
4. Round key generation loop:
5. Repeat the following steps for 16 rounds:
   1. Left shift:
      1. Shift C0 and D0 left by 1 or 2 bits according to the round number (specified in the DES specification).
   2. Permutation (PC2):
      1. Combine C0 and D0 back into a 56-bit key.
      2. Apply the PC2 (Permuted Choice 2) table to select 48 bits from the 56-bit key, forming the subkey for this round.
   3. Output:
      1. Display or store the generated 48-bit subkey.
6. Output:
   1. After 16 rounds, you'll have 16 distinct 48-bit subkeys.

**Source Code:**

#include<iostream>

#include<string>

#include<bitset>

using namespace std;

string round\_keys[16];

// circular left shift by one

string C\_L\_Shift\_Once(string key\_chunk)

{

string shifted = "";

for (int i = 1; i < 28; i++)

{

shifted += key\_chunk[i];

}

shifted += key\_chunk[0];

return shifted;

}

// circular left shift by two

string C\_L\_Shift\_Twice(string key\_chunk)

{

string shifted = "";

for (int i = 0; i < 2; i++)

{

for (int j = 1; j < 28; j++)

{

shifted += key\_chunk[j];

}

shifted += key\_chunk[0];

key\_chunk = shifted;

shifted = "";

}

return key\_chunk;

}

void key\_generate(string key)

{

// initial permutation table to convert the key in 56bits

int ip[56] = {

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

};

// compression permutation table to compress the key in 48bits

int cp[48] = {

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

};

// compressing the Key to 56 bit using compression permutation table

string perm\_key ="";

for(int i = 0; i < 56; i++)

{

perm\_key+= key[ip[i]-1];

}

// dividing the the 56 key into two part

string left = perm\_key.substr(0, 28);

string right = perm\_key.substr(28, 56);

// generating 16 round key

for (int i = 0; i < 16; i++)

{

// one left circular for 1, 2, 9, 16

if (i == 0 || i == 1|| i == 8 || i == 15)

{

left = C\_L\_Shift\_Once(left);

right = C\_L\_Shift\_Once(right);

}

else

{

left = C\_L\_Shift\_Twice(left);

right = C\_L\_Shift\_Twice(right);

}

// key chunks are combined

string combined\_key = left + right;

string round\_key = "";

for (int i = 0; i < 48; i++)

{

round\_key += combined\_key[cp[i]-1];

}

round\_keys[i] = round\_key;

cout << "Key "<< i+1 << ":" << round\_keys[i] << endl;

}

}

string TextToBinaryString(string words)

{

string binaryString = "";

for (char& \_char : words) {

binaryString +=bitset<8>(\_char).to\_string();

}

return binaryString;

}

int main()

{

string key, Plain\_Text, key\_bin;

cout << "Enter the key to encrypt" << endl;

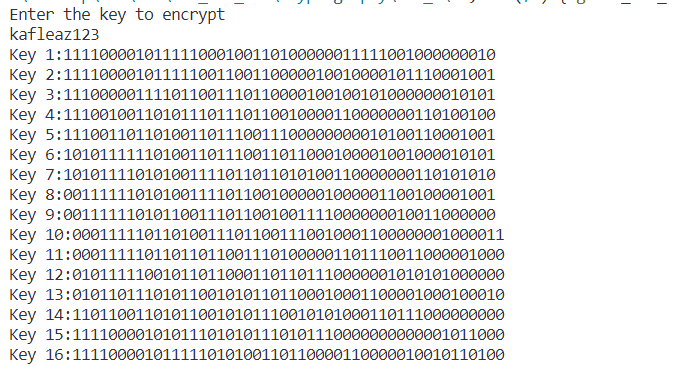
cin >> key;

key\_bin = TextToBinaryString(key).substr(0, 64);

key\_generate(key\_bin);

}

**Output:**



**Lab 2.2: Write a program to apply the round function to a given 32-bit data and subkey, and display the intermediate results.**

**Algorithm:**

DES key generation process to generate subkeys:

**Source Code:**