Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

Implementation Of DES

Aim: To study and implement DES algorithm

Theory:

The Data Encryption Standard (DES) is a symmetric-key block cipher algorithm that was developed in the early 1970s by IBM and adopted by the U.S. National Institute of Standards and Technology (NIST) as a federal standard in 1977. DES was widely used for secure data transmission and storage, though it has been largely replaced by more advanced encryption algorithms like the Advanced Encryption Standard (AES) due to its vulnerability to modern cryptographic attacks. Here's a brief overview of the theory behind the DES algorithm:

1. Key Generation:

- DES uses a 56-bit key for encryption and decryption. The key generation process starts with a user-supplied 64-bit key, where the most significant (leftmost) bit of each byte is used for error-checking (parity) and is not part of the actual key material.

- The key is reduced to 56 bits by discarding the parity bits. The resulting 56-bit key is divided into 16 subkeys, each 48 bits in length, one for each round of the encryption process.

2. Initial Permutation (IP):

- The 64-bit plaintext is subjected to an initial permutation, where the bits are rearranged according to a fixed permutation table.

3. Feistel Network:

- DES uses a Feistel network structure for encryption, which involves multiple rounds (16 rounds in the case of DES). In each round, the right half of the data is subjected to a function that depends on the round's subkey and the result is XORed with the left half.

- The left and right halves are swapped at the end of each round, and this process is repeated for a total of 16 rounds.

4. Subkey Mixing (F-function):

- The F-function takes a 32-bit half-block as input and expands it to 48 bits using the current round's subkey.

- The 48-bit result is then subjected to a series of operations, including substitution (S-boxes), permutation (P-box), and XOR operations with the 48-bit subkey.

- The output is a 32-bit value, which is then XORed with the other half of the data.

5. S-Boxes:

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

- DES employs eight 6x4-bit substitution boxes (S-boxes) to provide non-linear transformations in the F-function. Each S-box takes 6 bits as input and outputs 4 bits.

6. P-Box:

- After the F-function is applied, a permutation (P-box) is used to shuffle the bits of the 32-bit half-block.

7. Final Permutation (FP):

- After all rounds are completed, a final permutation is applied to the data, which is the inverse of the initial permutation, to produce the ciphertext.

8. Decryption:

- Decryption in DES is essentially the same as encryption, but the subkeys are used in reverse order.

Code:

```
#include <bits/stdc++.h>
using namespace std;
// Array to hold 16 keys
string round_keys[16];
// String to hold the plain text
string pt;
// Function to convert a number in decimal to binary

string textToBinary(const string &text)
{
    string binary = "";
    for (char c : text)
    {
        binary += bitset<8>(c).to_string();
    }
    return binary;
}

string binaryToText(const string &binary)
{
    string text = "";
    for (size_t i = 0; i < binary.length(); i += 8)
    {
        string byte = binary.substr(i, 8);
        char c = static_cast<char>(bitset<8>(byte).to_ulong());
        text += c;
    }
    return text;
}
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
string convertDecimalToBinary(int decimal)
{
    string binary;
    while (decimal != 0)
    {
        binary = (decimal % 2 == 0 ? "0" : "1") + binary;
        decimal = decimal / 2;
    }
    while (binary.length() < 4)
    {
        binary = "0" + binary;
    }
    return binary;
}
// Function to convert a number in binary to decimal
int convertBinaryToDecimal(string binary)
{
    int decimal = 0;
    int counter = 0;
    int size = binary.length();
    for (int i = size - 1; i >= 0; i--)
    {
        if (binary[i] == '1')
        {
            decimal += pow(2, counter);
        }
        counter++;
    }
    return decimal;
}
// Function to do a circular left shift by 1
string shift_left_once(string key_chunk)
{
    string shifted = "";
    for (int i = 1; i < 28; i++)
    {
        shifted += key_chunk[i];
    }
    shifted += key_chunk[0];
    return shifted;
}
// Function to do a circular left shift by 2
string shift_left_twice(string key_chunk)
{
    string shifted = "";
    for (int i = 0; i < 2; i++)
    {
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
        for (int j = 1; j < 28; j++)
        {
            shifted += key_chunk[j];
        }
        shifted += key_chunk[0];
        key_chunk = shifted;
        shifted = "";
    }
    return key_chunk;
}
// Function to compute xor between two strings
string Xor(string a, string b)
{
    string result = "";
    int size = b.size();
    for (int i = 0; i < size; i++)
    {
        if (a[i] != b[i])
        {
            result += "1";
        }
        else
        {
            result += "0";
        }
    }
    return result;
}
// Function to generate the 16 keys.
void generate_keys(string key)
{
    // The PC1 table
    int pc1[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};
    // The PC2 table
    int pc2[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,
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32};
    // 1. Compressing the key using the PC1 table
    string perm_key = "";
    for (int i = 0; i < 56; i++)
    {
        perm_key += key[pc1[i] - 1];
    }
    // 2. Dividing the key into two equal halves
    string left = perm_key.substr(0, 28);
    string right = perm_key.substr(28, 28);
    for (int i = 0; i < 16; i++)
    {
        // 3.1. For rounds 1, 2, 9, 16 the key_chunks
        // are shifted by one.
        if (i == 0 || i == 1 || i == 8 || i == 15)
        {
            left = shift_left_once(left);
            right = shift_left_once(right);
        }
        // 3.2. For other rounds, the key_chunks
        // are shifted by two
        else
        {
            left = shift_left_twice(left);
            right = shift_left_twice(right);
        }
        // Combining the two chunks
        string combined_key = left + right;
        string round_key = "";
        // Finally, using the PC2 table to transpose the key bits
        for (int i = 0; i < 48; i++)
        {
            round_key += combined_key[pc2[i] - 1];
        }
        round_keys[i] = round_key;
    }
}
// Implementing the algorithm
string DES()
{
    // The initial permutation table
    int initial_permutation[64] = {
        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,
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
        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};
    // The expansion table
    int expansion_table[48] = {
        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};
    // The substitution boxes. The should contain values
    // from 0 to 15 in any order.
    int substition_boxes[8][4][16] =
        {{14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
          0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
          4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
          15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13},
         {15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
          3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
          0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
          13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9},
         {10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
          13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
          13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
          1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12},
         {7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
          13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
          10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
          3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14},
         {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},
         {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},
         {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},
         {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}};
    // The permutation table
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
    int permutation_tab[32] = {
        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};
    // The inverse permutation table
    int inverse_permutation[64] = {
        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};
    // 1. Applying the initial permutation
    string perm = "";
    for (int i = 0; i < 64; i++)
    {
        perm += pt[initial_permutation[i] - 1];
    }
    // 2. Dividing the result into two equal halves
    string left = perm.substr(0, 32);
    string right = perm.substr(32, 32);
    // The plain text is encrypted 16 times
    for (int i = 0; i < 16; i++)
    {
        string right_expanded = "";
        // 3.1. The right half of the plain text is expanded
        for (int i = 0; i < 48; i++)
        {
            right_expanded += right[expansion_table[i] - 1];
        }; // 3.3. The result is xored with a key
        string xored = Xor(round_keys[i], right_expanded);
        string res = "";
        // 3.4. The result is divided into 8 equal parts and passed
        // through 8 substitution boxes. After passing through a
        // substituion box, each box is reduces from 6 to 4 bits.
        for (int i = 0; i < 8; i++)
        {
            // Finding row and column indices to lookup the
            // substituition box
            string row1 = xored.substr(i * 6, 1) + xored.substr(i * 6 + 5, 1);
            int row = convertBinaryToDecimal(row1);
            string col1 = xored.substr(i * 6 + 1, 1) + xored.substr(i * 6 + 2,
1) + xored.substr(i * 6 + 3, 1) + xored.substr(i * 6 + 4, 1);
            ;
            int col = convertBinaryToDecimal(col1);
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
            int val = substition_boxes[i][row][col];
            res += convertDecimalToBinary(val);
        }
        // 3.5. Another permutation is applied
        string perm2 = "";
        for (int i = 0; i < 32; i++)
        {
            perm2 += res[permutation_tab[i] - 1];
        }
        // 3.6. The result is xored with the left half
        xored = Xor(perm2, left);
        // 3.7. The left and the right parts of the plain text are swapped
        left = xored;
        if (i < 15)
        {
            string temp = right;
            right = xored;
            left = temp;
        }
    }
    // 4. The halves of the plain text are applied
    string combined_text = left + right;
    string ciphertext = "";
    // The inverse of the initial permuttaion is applied
    for (int i = 0; i < 64; i++)
    {
        ciphertext += combined_text[inverse_permutation[i] - 1];
    }
    // And we finally get the cipher text
    return ciphertext;
}
int main()
{
    // A 64 bit key
    string key;
    cout << "Enter a 64 bit (8 letter) key: ";
    cin >> key;

    // A block of plain text of 64 bits
    cout << "Enter a 64 bit (8 letter) plain text: ";
    cin >> pt;

    key = textToBinary(key);
    pt = textToBinary(pt);

    string apt = pt;
    // Calling the function to generate 16 keys
    generate_keys(key);
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
    cout << "Plain text: " << pt << endl;
    // Applying the algo
    string ct = DES();
    cout << "Ciphertext: " << ct << endl;
    // Reversing the round_keys array for decryption
    int i = 15;
    int j = 0;
    string x=
"0110011101101111011100000110000101101100011101100110100101101100";
    cout<<x.size()<<endl;
    while (i > j)
    {
        string temp = round_keys[i];
        round_keys[i] = round_keys[j];
        round_keys[j] = temp;
        i--;
        j++;
    }
    pt = ct;
    string decrypted = DES();
    cout << "Decrypted text: " << binaryToText(decrypted) << endl;
    // Comapring the initial plain text with the decrypted text
    if (decrypted == apt)
    {
        cout << "Plain text encrypted and decrypted successfully." << endl;
    }
}
```

Screenshot:

```
PS F:\D drive\0Walchand_Sem7\CNS lab> cd "f:\D drive\0Walchand_Sem7\CNS lab\DES_A
o DES } ; if ($?) { .\DES }
Enter a 64 bit (8 letter) key: 12345678
Enter a 64 bit (8 letter) plain text: 98765432
Plain text: 0011100100111000001101110011011000110101001101000011001100110010
Ciphertext: 1000011100000111011001010101101001010111110011110110000001011001
64
Decrypted text: 98765432
Plain text encrypted and decrypted successfully.
```

Name: Shreeshail Mahajan
PRN: 2020BTECS00055
Batch: B4

```
Encryption
After initial permutation 14A7D67818CA18AD
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
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
Cipher Text :  C0B7A8D05F3A829C
```

```
Decryption
After initial permutation 19BA9212CF26B472
Round  1   CF26B472   BD2DD2AB   181C5D75C66D
Round  2   BD2DD2AB   387CCDAA   3330C5D9A36D
Round  3   387CCDAA   22A5963B   251B8BC717D0
Round  4   22A5963B   FF3C485F   99C31397C91F
Round  5   FF3C485F   6CA6CB20   C2C1E96A4BF3
Round  6   6CA6CB20   10AF9D37   6D5560AF7CA5
Round  7   10AF9D37   308BEE97   02765708B5BF
Round  8   308BEE97   A9FC20A3   84BB4473DCCC
Round  9   A9FC20A3   2E8F9C65   34F822F0C66D
Round  10   2E8F9C65   A15A4B87   708AD2DDB3C0
Round  11   A15A4B87   236779C2   C1948E87475E
Round  12   236779C2   B8089591   69A629FEC913
Round  13   B8089591   4A1210F6   DA2D032B6EE3
Round  14   4A1210F6   5A78E394   06EDA4ACF5B5
Round  15   5A78E394   18CA18AD   4568581ABCCE
Round  16   14A7D678   18CA18AD   194CD072DE8C
Plain Text :  123456ABCD132536
```