**Submitted by – Paras Jain (2018kucp1006) ISS Lab**

**Assignment: Implementation of different Modes Of operation**

Here I have implemented 4 different modes of operations on my DES algorithm code.

1. Cipher Block Chaining Mode (CBC Mode)
2. Cipher Feedback Mode (CFB Mode)
3. Output Feedback Mode (OFB Mode)
4. Counter Mode (CTR Mode)

Code:

#include <bits/stdc++.h>

using namespace std;

typedef long long lint;

// This function converts from decimal to binary

string toBin(int temp){

    string ans;

    while (temp > 0)

    {

        if (temp % 2 == 0)

            ans.push\_back('0');

        else

            ans.push\_back('1');

        temp /= 2;

    }

    reverse(ans.begin(), ans.end());

    return ans;

}

// this function converts plain text to binary sequence according to ascii values

string chToBin(string ptxt){

    string ans = "";

    for (int i = 0; i < ptxt.length(); i++){

        int temp = ptxt[i];

        ans.push\_back('0');

        string temp\_ans;

        while (temp > 0)

        {

            if (temp % 2 == 0)

                temp\_ans.push\_back('0');

            else

                temp\_ans.push\_back('1');

            temp /= 2;

        }

        reverse(temp\_ans.begin(), temp\_ans.end());

        ans += temp\_ans;

    }

    return ans;

}

// This function converts binary string to decimal

int toDec(string s){

    int ans = 0, base = 1;

    for (int i = 0; i < s.length(); i++)

    {

        ans \*= 2;

        if (s[i] == '1')

            ans += 1;

    }

    return ans;

}

// to perform initial permutation on plain txt

string iniPermut(string ptxt)

{

    int choice[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,

        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};

    string t1;

    for (int i = 0; i < 64; i++)

    {

        t1.push\_back(ptxt[choice[i] - 1]);

    }

    return t1;

}

// to apply permutation choice 1 on the key

string pc1(string key){

    int choice[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};

    string t1;

    for (int i = 0; i < 56; i++)

    {

        t1.push\_back(key[choice[i] - 1]);

    }

    return t1;

}

//Left Shift by 1

void leftShift1(string &key)

{

    string c, d;

    for (int i = 1; i < 28; i++)

    {

        c.push\_back(key[i]);

    }

    c.push\_back(key[0]);

    for (int i = 29; i < 56; i++)

    {

        d.push\_back(key[i]);

    }

    d.push\_back(key[28]);

    key = c + d;

}

// Left Shift by 2

void leftShift2(string &key)

{

    string c, d;

    for (int i = 2; i < 28; i++)

    {

        c.push\_back(key[i]);

    }

    c.push\_back(key[0]);

    c.push\_back(key[1]);

    for (int i = 30; i < 56; i++)

    {

        d.push\_back(key[i]);

    }

    d.push\_back(key[28]);

    d.push\_back(key[29]);

    key = c + d;

}

// the below table is chosen from the slides

// provided to us during the class

string pc2(string key)

{

    int choice[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};

    string t1;

    for (int i = 0; i < 48; i++)

    {

        t1.push\_back(key[choice[i] - 1]);

    }

    key = t1;

    return key;

}

// the below table is chosen from the slides

// provided to us during the class

void expBox(string &r)

{

    int choice[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};

    string t1;

    for (int i = 0; i < 48; i++)

    {

        t1.push\_back(r[choice[i] - 1]);

    }

    r = t1;

}

// the below tables are chosen from slides and the book

void sBox(string &r)

{

    int s1[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};

    int s2[4][16] = {

        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};

    int s3[4][16] =

        {

            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};

    int s4[4][16] =

        {

            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};

    int s5[4][16] =

        {

            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};

    int s6[4][16] =

        {

            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};

    int s7[4][16] =

        {

            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};

    int s8[4][16] =

        {

            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};

    string t1 = "";

    for (int i = 0; i < r.size(); i += 6)

    { //r.size is 48

        string row, col;

        row.push\_back(r[i]);

        row.push\_back(r[i + 5]);

        col.push\_back(r[i + 1]);

        col.push\_back(r[i + 2]);

        col.push\_back(r[i + 3]);

        col.push\_back(r[i + 4]);

        int rn = toDec(row);

        int cn = toDec(col);

        int contr;

        if (i == 0)

            contr = s1[rn][cn];

        else if (i == 6)

            contr = s2[rn][cn];

        else if (i == 12)

            contr = s3[rn][cn];

        else if (i == 18)

            contr = s4[rn][cn];

        else if (i == 24)

            contr = s5[rn][cn];

        else if (i == 30)

            contr = s6[rn][cn];

        else if (i == 36)

            contr = s7[rn][cn];

        else if (i == 42)

            contr = s8[rn][cn];

        string s1 = toBin(contr);

        int diff = 4 - s1.size();

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

            t1.push\_back('0');

        t1 += s1;

    }

    r = t1;

}

// the below table is chosen from the slides

// provided to us during the class

void permute(string &r)

{

    int choice[] = {

        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};

    string t1;

    for (int i = 0; i < 32; i++)

    {

        t1.push\_back(r[choice[i] - 1]);

    }

    r = t1;

}

// Each round of des

void round(string &key, string &l, string &r)

{

    // making a copy of right portion

    string r\_temp = r;

    // expansion

    expBox(r);

    //now xor with key

    for (int i = 0; i < 48; i++)

    {

        if (r[i] == key[i])

            r[i] = '0';

        else

            r[i] = '1';

    }

    // s - boxing

    sBox(r);

    // permutation

    permute(r);

    //now xor with left portion

    for (int i = 0; i < 32; i++)

    {

        if (r[i] == l[i])

            r[i] = '0';

        else

            r[i] = '1';

    }

    // using the copy of right portion to make the

    // left portion of th next round

    l = r\_temp;

}

// Invert initial permutation

void invPermut(string &cipher)

{

    int choice[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,

        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};

    string t1;

    for (int i = 0; i < 64; i++)

    {

        t1.push\_back(cipher[choice[i] - 1]);

    }

    cipher = t1;

}

string BinToStr(string s){

    string tmp, ans;

    for (int i = 0; i < s.size(); i++){

        if (i != 0 and i % 8 == 0){

            char d = toDec(tmp);

            tmp.clear();

            ans.push\_back(d);

        }

        tmp.push\_back(s[i]);

    }

    char d = toDec(tmp);

    ans.push\_back(d);

    return ans;

}

string DES(string ptxt, string key){

    ptxt = chToBin(ptxt);

    //cout << "Initial Plain txt bits = " << ptxt << endl;

    ptxt = iniPermut(ptxt);

    key = chToBin(key);

    //cout << "Initial Key txt bits = " << key << endl;

    key = pc1(key);

    string cipher;

    vector<string> key\_vect;

    for (int i = 1; i <= 16; i++)

    {

        if (i == 1 or i == 2 or i == 9 or i == 16)

        {

            leftShift1(key);

        }

        else

            leftShift2(key);

        string temp\_key = pc2(key); //48-bit keys

        key\_vect.push\_back(temp\_key);

        string l, r;

        for (int i = 0; i < 32; i++)

        {

            l.push\_back(ptxt[i]);

            r.push\_back(ptxt[i + 32]);

        }

        round(temp\_key, l, r);

        ptxt = l + r;

        //cout << "cipher bits after round " << i << " = " << ptxt << endl;

    }

    // inverting initial permutation

    invPermut(ptxt);

    string s = BinToStr(ptxt);

    return s;

}

string iv = "CONTACTS"; //Initilization Vector

//Cipher Block Chaining Mode

void CBC(string &str){

    string temp;

    for(int i=0;i<8;i++){

        char t = iv[i]^str[i];

        temp.push\_back(t);

    }

    str = temp;

    iv = str;

}

void CBC\_main(string pt, string key)

{

    iv = "CONTACTS";

    for (int i = 0; i < pt.length() / 8; i++)

    {

        string ptxt;

        for (int j = i \* 8; j < i \* 8 + 8; j++)

        {

            ptxt.push\_back(pt[j]);

        }

        CBC(ptxt);

        cout<<DES(ptxt, key);

    }

}

//Cipher Feedback Mode

string CFM(string ptxt, string o, int round=0){

    string temp;

    int k=0;

    for(int i=round\*4;i<round\*4 + 4;i++){

        temp.push\_back(ptxt[i]^o[k]);

        k++;

    }

    string temp\_iv;

    for(int i=3;i<8;i++){

        temp\_iv.push\_back(iv[i]);

    }

    for(int i=0;i<4;i++){

        temp\_iv.push\_back(temp[i]);

    }

    iv = temp\_iv;

    return temp;

}

void CFM\_main(string pt, string key)

{

    iv = "CONTACTS";

    for (int i = 0; i < pt.length() / 4; i++)

    {

        string ptxt;

        for (int j = i \* 4; j < i \* 4 + 4; j++)

        {

            ptxt.push\_back(pt[j]);

        }

        string o = DES(iv, key);

        cout<<CFM(ptxt,o,i);

    }

}

//Output Feedback Mode

string OFM(string ptxt, string o, int round=0){

    string temp;

    int k=0;

    for(int i=round\*8;i<round\*8 + 8;i++){

        temp.push\_back(ptxt[i]^o[k]);

        k++;

    }

    iv = o;

    return temp;

}

void OFM\_main(string pt, string key)

{

    iv = "CONTACTS";

    for (int i = 0; i < pt.length() / 8; i++)

    {

        string ptxt;

        for (int j = i \* 8; j < i \* 8 + 8; j++)

        {

            ptxt.push\_back(pt[j]);

        }

        string o = DES(iv, key);

        cout<<OFM(ptxt,o,i);

    }

}

//Counter Mode

void CTR(string &str,string ctr)

{

    string temp;

    for (int i = 0; i < 8; i++)

    {

        char t = ctr[i] ^ str[i];

        temp.push\_back(t);

    }

    str = temp;

}

void CTR\_main(string pt,string key)

{

    string ctr = "CONTACTS";

    for (int i = 0; i < pt.length() / 8; i++)

    {

        string ptxt;

        for (int j = i \* 8; j < i \* 8 + 8; j++)

        {

            ptxt.push\_back(pt[j]);

        }

        CTR(ptxt,ctr);

        cout << DES(ptxt, key);

        ctr[7]++;

    }

}

int main(){

    string pt = "parasJainID\_1006", key = "MONARCHY";

    cout<<"Plain Text: "<<pt<<"\nKey: "<<key<<"\n";

    cout<<"Output in CBC mode: ";

    CBC\_main(pt,key);

    cout<<"\nOutput in CFM mode: ";

    CFM\_main(pt, key);

    cout<<"\nOutput in OFM mode: ";

    OFM\_main(pt, key);

    cout<<"\nOutput in CTR mode: ";

    CTR\_main(pt, key);

}

Output:

