1_monoalphabetic_cipher.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: MonoAlphabetic cipher
Author: Gangadhara Shetty P J

Algorithm
---------
A mono-alphabetic cipher is a type of simple substitution cipher. In this cipher technique each letter of
the plaintext is replaced by another letter in the cipher-text. An example of a mono-alphabetic cipher key follows:

Plain Text  >>>   a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
Cipher Text >>>  z  w  x  y  o  p  r  q  a  h  c  b  e  s  u  t  v  f  g  j  l  k  m  n  d  i

This key means that any ‘a’ in the plaintext will be replaced by a ‘z’ in the cipher-text, any ‘z’ in the plaintext will be replaced by a ‘i’ in the cipher-text, and so on.
The following program shows the simple implementation of mono-alphabetic cipher technique in c language for encrypting and decrypting file

*/
#include<bits/stdc++.h>

using namespace std;

int frequency_count[256];
vector<string> keys;

// To encrypt the plain_text with the key
string encrypt(string plain_text, string unique_characters, string key)
{
    for(int i=0; i<plain_text.length(); i++)
        plain_text[i] = key[ unique_characters.find( plain_text[i] ) ];
    return plain_text;
}

void generateKeysByPermutation(string text)
{
    sort( text.begin(), text.end());
    // next_permutation is a readily available function to generate permutation of the string
    while(next_permutation(text.begin(), text.end()))
    {
//          cout<<text<<endl;
        keys.push_back(text);
    }
}

string readFile(char* file_name)
{
    string text;
    ifstream fin(file_name);
    fin>>text;
    fin.close();
    return text;
}

void storeFile(char* file_name, string text)
{
    ofstream fout(file_name);
    fout<<text;
    fout.close();
}

string getUniqueCharacters(string plain_text)
{
    int flag[255]={0};
    string unique_characters="";
        // To get the list of unique characters in the above string
        // Also get the number of occurence of the characters to find the frequence later
    for(int i=0;i<plain_text.length();i++){
        if( !frequency_count[plain_text[i]] )
            unique_characters+=plain_text[i];
                frequency_count[plain_text[i]]++;
        }
    return unique_characters;
}

void calculateFrequency(string unique_characters, string key, string text)
{
    float length = text.length();
    cout<<"Frequency\tUnique Characters\tChoosen Key"<<endl;
    cout<<"------------------------------------------------------------------"<<endl;
        // Frequency / Length * => percentage of occurance of the character in the entire string
        for(int i=0; i<unique_characters.length();i++)
                cout << frequency_count[ unique_characters[i] ] / length * 100 << "\t\t\t" << unique_characters[i] << "\t\t" << key[i] << endl;
}

int main() {
        string unique_characters, key, plain_text, cipher_text;

    srand(time(0));

    plain_text = readFile("plain_text.txt");

    cout<<"plain_text:\t"<<plain_text<<endl;

    unique_characters = getUniqueCharacters( plain_text );
        cout<<"unique character:\t"<<unique_characters<<endl;

    generateKeysByPermutation( unique_characters );
    key = keys[rand()%keys.size()];
    cout<<"Choosen Key:\t"<<key<<endl;

    cipher_text = encrypt( plain_text, unique_characters, key );
    cout<<"cipher_text:\t"<<cipher_text<<endl;

    storeFile("cipher_text.txt", cipher_text);

    calculateFrequency(unique_characters, key, plain_text);

        return 0;
}

/*
-------------
Sample Output
-------------

plain_text:     gauggsdgssgsgsasuuasudduugdsugs
unique character:       gausd
Choosen Key:    sgaud
```

```
cipher_text:    sgassudsuusususguaaguaddaasduasu

Frequency       Unique Characters       Choosen Key
-----------------------------------------------------------------
25.8065                 g               s
9.67742                 a               g
22.5806                 u               a
29.0323                 s               u
12.9032                 d               d
*/
```

2_play_fair_cipher.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: Playfair cipher
Author: Gangadhara Shetty P J

Algorithm
---------
    The 'key' for a playfair cipher is generally a word, for the sake of example we will choose 'monarchy'.
        This is then used to generate a 'key square'. Any sequence of 25 letters can be used as a key,
        so long as all letters are in it and there are no repeats. Note that there is no 'j', it is combined with 'i'.
        We now apply the encryption rules to encrypt the plaintext.

    1. Remove any punctuation or characters that are not present in the key square (this may mean spelling out numbers, punctuation etc.).
    2. Identify any double letters in the plaintext and replace the second occurence with an 'x' e.g. 'hammer' -> 'hamxmerx'.
    3. If the plaintext has an odd number of characters, append an 'x' to the end to make it even.
    4. Break the plaintext into pairs of letters, e.g. 'hamxer' -> 'ha mx er'
    5. The algorithm now works on each of the letter pairs.
    6. Locate the letters in the key square, (the examples given are using the key square above)

        a. If the letters are in different rows and columns, replace the pair with the letters on the same row respectively
                   but at the other pair of corners of the rectangle defined by the original pair.
                   The order is important – the first encrypted letter of the pair is the one that lies on the same row as the first plaintext letter.

        b. If the letters appear on the same row of the table, replace them with the letters to their immediate right respectively
                   (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).

        c. If the letters appear on the same column of the table, replace them with the letters immediately below respectively
                   (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).

*/

#include<iostream>
#include<cmath>

using namespace std;

char matrix[5][5];

// Generate a matrix from the given key.
// Assuming all 'j' as 'i', since they can be used interchangably.
// Remove spaces in a key.
void GenerateMatrix(string key)
{
    int flag[26]={0};
    int x_ind=0, y_ind=0;
    // Add all characters present in a given key
    for(int i=0;i<key.length();i++)
    {
        if(key[i]==' ') continue;
        if(key[i]=='j') key[i]='i';

        if(!flag[key[i]-97])
            matrix[x_ind][y_ind++] = key[i], flag[key[i]-97]=1;

        if(y_ind==5)    x_ind++, y_ind=0;
    }

    // Add all other characters
    for(int i=0;i<26;i++)
    {
        if(i==9)    continue;
        if(key[i]=='j') key[i]='i';
        if(!flag[i])    matrix[x_ind][y_ind++] = i+97, flag[i]=1 ;

        if(y_ind==5)    x_ind++, y_ind=0;
    }
}

string FormatMessage(string message)
{
    // STEP 1 in algorithm
    for(int i=0;i<message.length();i++)
    {
        if(message[i] == ' ')   // add conditions here for any other special characters
            message = message.replace(i, 1, "");
        if(message[i] == 'j')
            message = message.replace(i, 1, "i");
    }

    // STEP 2 in algorithm
    for(int i=1;i<message.length();i+=2)
         if(message[i-1] == message[i])
            message = message.insert(i, "x");
    // STEP 3 in algorithm
    if(message.length()%2)
        message += "x";
    return message;
}

// Returns the row position of the given character in a matrix
int GetRow(char c)
{
    for(int i=0;i<5;i++)
        for(int j=0;j<5;j++)
            if(c==matrix[i][j])
                return i;
}

//Returns the column position of the given character in a matrix
int GetColumn(char c)
{
    for(int i=0;i<5;i++)
        for(int j=0;j<5;j++)
            if(c==matrix[i][j])
                return j;
}


string Encrypt(string message)
{
    string enc_msg;
    for(int i=0;i<message.length();i+=2)    // i is incremented by 2 inorder to group by
two two characters
```

```cpp
    {
        int xind1 = GetRow(message[i]);
        int xind2 = GetRow(message[i+1]);
        int yind1 = GetColumn(message[i]);
        int yind2 = GetColumn(message[i+1]);

        // STEP 6.a in algoritm
        if( xind1 == xind2 )
        {
            enc_msg.append(1, matrix[xind1][(yind1+1)%5]);
            enc_msg.append(1, matrix[xind2][(yind2+1)%5]);
        }
        //STEP 6.b in algorithm
        else if( yind1 == yind2 )
        {
            enc_msg.append(1, matrix[(xind1+1)%5][yind1]);
            enc_msg.append(1, matrix[(xind2+1)%5][yind2]);
        }
        // STEP 6.c in algorithm
        else
        {
            enc_msg.append(1, matrix[ xind1 ][ yind2 ]);
            enc_msg.append(1, matrix[ xind2 ][ yind1 ]);
        }
    }
    return enc_msg;
}


string Decrypt(string message)
{
    string msg;
    for(int i=0;i<message.length();i+=2)
    {
        int xind1 = GetRow(message[i]);
        int xind2 = GetRow(message[i+1]);
        int yind1 = GetColumn(message[i]);
        int yind2 = GetColumn(message[i+1]);

        // STEP 6.a in algoritm (reveres)
        if( xind1 == xind2 )
        {
            msg.append(1, matrix[xind1][ --yind1<0 ? 4: yind1 ]);    // to handle
negative modlus  if(negative) 4 else num-1
            msg.append(1, matrix[xind2][ --yind2<0 ? 4: yind2 ]);
        }
        // STEP 6.b in algoritm (reverse)
        else if( yind1 == yind2 )
        {
            msg.append(1, matrix[ --xind1<0 ? 4: xind1 ][yind1]);
            msg.append(1, matrix[ --xind2<0 ? 4: xind2 ][yind2]);
        }
        // STEP 6.c in algoritm (reverse)
        else
        {
            msg.append(1, matrix[ xind1 ][ yind2 ]);
            msg.append(1, matrix[ xind2 ][ yind1 ]);
        }
    }
    return msg;
}

int main()
{
        string message, keys[100];
        int  num_of_keys;

        cout<<"Enter the number of keys:";
        cin>>num_of_keys;
        cin.get();

        cout<<"Enter the keys:";
        for(int i=0;i<num_of_keys;i++)
            getline(cin, keys[i]);

    cout<<"Enter a message to be encrypted: ";
    getline(cin, message);

    for(int i=0;i<num_of_keys;i++)
    {
        GenerateMatrix(keys[i]);
        cout<<endl<<endl;
        cout<<"----------------------------"<<endl;
        cout<<"Using key"<<i<<": "<<keys[i]<<endl;
        cout<<"----------------------------"<<endl;
        cout<<"Key Matrix: "<<endl;
        for(int k=0;k<5;k++)
        {
            for(int j=0;j<5;j++)
                cout<<matrix[k][j]<<" ";
            cout<<endl;
        }
        cout<<"Actual Message: "<<message<<endl;
        string for_msg = FormatMessage(message);
        cout<<"Formatted Message: "<<for_msg<<endl;
        string enc_msg = Encrypt(for_msg);
        cout<<"Encrypted Message: "<<enc_msg<<endl;
        string dec_msg = Decrypt(enc_msg);
        cout<<"Decrypted Message: "<<dec_msg<<endl;
    }
        return 0;
}

/*
-------------
Sample Output
-------------

Enter the number of keys:1
Enter the keys:playfair example
Enter a message to be encrypted: hide the gold

----------------------------
Using key0: playfair example
----------------------------
Key Matrix:
p l a y f
i r e x m
b c d g h
k n o q s
t u v w z
Actual Message: hide the gold
Formatted Message: hidethegoldx
Encrypted Message: bmodzbxdnage
Decrypted Message: hidethegoldx
*/
```

3_hill_cipher.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: Hill cipher
Author: Gangadhara Shetty P J
B
Algorithm
----------
   Hill cipher is a polygraphic substitution cipher based on linear algebra.Each letter is represented by a number modulo 26.
   Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher.
   To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26.
   To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

        The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).

*/

#include<bits/stdc++.h>

using namespace std;

int keyMatrix[100][100], inverseMatrix[100][100];
int order;

string FormatMessage(string message)
{
    for(int i=0;i<message.length();i++)
        if(message[i] == ' ')   // add conditions here for any other special characters
            message = message.replace(i, 1, "");


    for(int i=1;i<message.length();i++)
         if(message[i-1] == message[i])
            message = message.insert(i, "x"), i++;

    if(message.length()%order)
        message += "x";
    return message;
}

int GetInverseDeterminant(int R , int D = 26){ //R is the remainder or determinant
        int i =0 ;
        int p0= 0 , p1 =1 ;
        int q = 1 ;
        int q0 , q1 ;
        while(R!=0){
                q = D/R ;
                int tempD = D ;
                D = R ;
                R = tempD%R ;

                if(i==0) { p0 = 0 ; q0 = q ; }
                else if(i==1){ p1==1 ;q1 = q ; }
                else{
                        int temp = p1 ;
                        p1 = (p0-p1*(q0))%26 ;
                        if(p1<0)p1 = 26-(abs(p1)%26) ;
                        p0 = temp ;

                        q0 = q1;
                        q1 = q ;
                }
//              cout<<"p0 , p1 = " << p0 <<" " <<p1<< endl;

                i++ ;
        }
        p1 = (p0-p1*(q0))%26 ;
        return p1 ;
}

int GetDeterminant()
{
    int determinant = 0;
    if(order==2)
        determinant = keyMatrix[0][0] * keyMatrix[1][1] - keyMatrix[0][1] * keyMatrix[1][0];
    else
        for(int i = 0; i < 3; i++)
            determinant = determinant + (keyMatrix[0][i] * (keyMatrix[1][(i+1)%3] * keyMatrix[2][(i+2)%3] - keyMatrix[1][(i+2)%3] * keyMatrix[2][(i+1)%3]));

    if(determinant<0)
        determinant = 26 - (int(-determinant)%26);
    else
        determinant = int(determinant)%26;
    determinant = GetInverseDeterminant(determinant, 26);

    return determinant;
}

string Multiply(string msg_group, int matrix[][100])
{
    string result;
    for(int i=0; i<order; i++)
    {
        float val = 0 ;
        for(int j=0;j<order; j++)
            val = val + matrix[j][i] * (msg_group[j] - 'a');

        if(val>=0)
            val =  int(val)%26 + 'a';
        else
            val = 26 - (int(-val)%26) + 'a';

        result += int(val);
    }
    return result;
}

void FindInverse(int determinant)
{
    if(order==2)
    {
        inverseMatrix[0][0] = keyMatrix[1][1]*determinant;
        inverseMatrix[1][1] = keyMatrix[0][0]*determinant;
        inverseMatrix[0][1] = -keyMatrix[0][1]*determinant;
        inverseMatrix[1][0] = -keyMatrix[1][0]*determinant;
    }
    else
        for(int j=0; j<order; j++)
            for(int i=0; i<order; i++)
                inverseMatrix[i][j] = ((keyMatrix[(j+1)%3][(i+1)%3] * keyMatrix[(j+2)%3][(i+2)%3]) - (keyMatrix[(j+1)%3][(i+2)%3] * keyMatrix[(j+2)%3][(i+1)%3]))*determinant;
```

```
         for(int j=0; j<order; j++)
            for(int i=0; i<order; i++)
                if(inverseMatrix[i][j] < 0 )    inverseMatrix[i][j] = 26 - int(-
inverseMatrix[i][j])%26;
                                else inverseMatrix[i][j] = int(inverseMatrix[i][j]) %26;
}

string Encrypt(string message)
{
    string enc_msg;
    for(int i=0;i<message.length();i+=order)
    {
        string msg_group = message.substr(i, order);
        msg_group = Multiply(msg_group, keyMatrix);
        enc_msg = enc_msg.append(msg_group);
    }
    return enc_msg;
}

string Decrypt(string message)
{
    string msg;
    FindInverse(GetDeterminant());
    for(int i=0;i<message.length();i+=order)
    {
        string msg_group = message.substr(i, order);
        msg_group = Multiply(msg_group, inverseMatrix);
        msg = msg.append(msg_group);
    }
    return msg;
}

int main()
{
        string message;
    cout << "Enter the key matrix order: ";
    cin >> order;

        cout << "Enter the keys:";
        for(int i = 0; i < order; i++)
            for(int j=0;j<order; j++)
                cin >> keyMatrix[i][j];
    cin.get();

    cout<<"Enter a message to be encrypted: ";
    getline(cin, message);

    FormatMessage(message);
    string enc_msg = Encrypt(message);
    string dec_msg = Decrypt(enc_msg);
    cout<<"Message: "<<message<<endl;
    cout<<"Encrypted Message: "<<enc_msg<<endl;
    cout<<"Decrypted Message: "<<dec_msg<<endl;

        return 0;
}

/*
-------------
Sample Output
-------------
Enter the key matrix order: 3
Enter the keys:17 17 5
21 18 21
2 2 19
```

```
Enter a message to be encrypted: paymoremoney
Message: paymoremoney
Encrypted Message: rrlmwbkaspdh
Decrypted Message: paymoremoney

*/
```

4_transposition_cipher.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: Columnar Transposition Cipher
Author: Gangadhara Shetty P J
Algorithm
---------
In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.
        1. The message is written out in rows of a fixed length, and then read out again column by column,
                and the columns are chosen in some scrambled order.
        2. Width of the rows and the permutation of the columns are usually defined by a keyword.
        3. Any spare spaces are filled with nulls or left blank or placed by a character to
                construct the rectangular matrix (Example: $).
        4. Finally, the message is read off in columns, in the order specified by the keyword.
*/

#include<bits/stdc++.h>

using namespace std;


string Encrypt(string message, string keys)
{
    string enc_msg;
    int key[keys.length()], i;

    for(int k=0;k<keys.length();k++)
        key[keys[k]-'0'-1] = k;

    for(int k=0;k<keys.length();k++){
        for(i=key[k]; i<message.length(); i+=keys.length())
            enc_msg+=message[i];
        if(i%keys.length() >= message.length()%keys.length())
            enc_msg+="$";
    }
    return enc_msg;
}

string Decrypt(string message, string keys)
{
    string msg;
    int key[keys.length()], i, len = message.length()/ keys.length();

    for(int k=0;k<len;k++)
        for(i=0;i<keys.length();i++)
            msg+=message[(keys[i]-'0'-1)*len + k];
    return msg;
}

int main()
{
        string message, keys;

    cout<<"Enter a message to be encrypted: ";
    getline(cin, message);

        cout << "Enter the keys as a string:";
        getline(cin, keys);

    string enc_msg = Encrypt(message, keys);
    string dec_msg = Decrypt(enc_msg, keys);
    cout<<"Encrypted Message: "<<enc_msg<<endl;
    cout<<"Decrypted Message: "<<dec_msg<<endl;

        return 0;
}

/*
-------------
Output
-------------
Enter a message to be encrypted: hidethegold
Enter the keys as a string:14253
Encrypted Message: hhddg$tl$ie$eo$
Decrypted Message: hidethegold$$$$
*/
```

keygen.cpp in gangadharashettypj/labs (master)

```cpp
/*
lab 5
Program: DES Key generator
Author: Gangadhara Shetty P J
*/
#include <bits/stdc++.h>
using namespace std;

int permChoiceOne[] = {
                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
        };

int permChoiceTwo[] = {
                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
        };

int leftShiftTable[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};
string rotateSubKey(string s , int rot)
{
        return s.substr(rot, s.length()-rot) + s.substr(0, rot) ;
}
string firstPermute(string input)
{
        string res = "" ;
        for(int i=0 ; i<56 ; i++)
                res += input[permChoiceOne[i]-1];
        return res ;
}
string secondPermute(string input)
{
        string res = "" ;
        for(int i=0 ; i<48 ; i++)
                res += input[permChoiceTwo[i]-1];
        return res ;
}
void genKeys(string left, string right)
{
        ofstream fout ;
        fout.open("keygen.txt");
        for (int i=0; i<16; i++)
        {
                left = rotateSubKey(left , leftShiftTable[i]);
                right = rotateSubKey(right, leftShiftTable[i]);
                string key = secondPermute(left+right);
                cout << "key " << i+1 << " \t: " << key << endl;
                fout << key << endl;
        }
}

```

```cpp
int main()
{
        unsigned long long hexkey;
        cout << "\nEnter key in hexadecimal : " ;
        cin >> hex >> hexkey; // to read hex input cin >> hex >> input
        string key = bitset<64>(hexkey).to_string(); // to convert hex to binary string
        cout << "Binary key (k) \t: " << key << endl;
        key = firstPermute(key) ;
        cout << "PC-1 key (k+) \t: " << key << endl;
        cout << "\nSubKeys: " << endl;
        genKeys(key.substr(0,28) , key.substr(28,28));
        cout<<endl<<endl ;
}

/*
Enter key in hexadecimal : 1FE22472901BB2A3
Binary key (k)  : 0001111111100010001001000111001010010000000110111011001010100011
PC-1 key (k+)   : 11010010000010101100111001111110101100000101001000011001

SubKeys:
key 1   : 110010110110001010101100110001101000000110100011
key 2   : 001101011010010010101111001001011110000110011010
key 3   : 111100110000110010000001001100101000101000100011
key 4   : 011110001010101010110100110011101000000001101110
key 5   : 100101001011010000011110000000100110111111001100
key 6   : 011001100000011001110110000110001001010011110001
key 7   : 111011101101100000100100110010111100110000100001
key 8   : 100010101010001101111010000010100110111100011000
key 9   : 111000111100001011010111100101000001110000110011
key 10  : 001111011101001110000010100011110000101001110100
key 11  : 001100100001000111111011000100011110101111010000
key 12  : 101111010100000001010101001100011000010000010101
key 13  : 000001110100101110011100110010110010010010000110
key 14  : 000111100011000110110101001011000110001110001101
key 15  : 100111110000110001101001001100100101000011000111
key 16  : 010010011010100110011011011010100010100101010110

*/
```

sbox input.cpp in gangadharashettypj/labs (master)

```cpp
/*
lab 6
Program: SBOX input generator
Author: Gangadhara Shetty P J

*/
#include<bits/stdc++.h>
using namespace std;

string key;
int permute[]={
    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
};
void findPermutation()
{
    string key1;
    for(int i=0;i<48;i++)
        key1 += key[permute[i]-1+32];
    key=key1;
}
int main()
{

    unsigned long long hexKey;
    string inputkey;
    cout<<"Enter a 64 bit key in hex: ";
    cin>>hex>>hexKey;
        key=bitset<64>(hexKey).to_string();
        findPermutation();
        cout<<"Enter a 48 bit input key in hex: ";
        cin>>hex>>hexKey;

        inputkey=bitset<48>(hexKey).to_string();

        for(int i=0;i<48;i++)
            if(key[i]==inputkey[i]) key[i]='0';
            else key[i]='1';

        cout<<"S-BOX INPUT: "<<hex<< bitset<48>(key).to_ulong()<<endl;

}
/*
OUTPUT
------
Enter a 64 bit key in hex: aaaaaaaaf0aaf0aa
Enter a 48 bit input key in hex: 1b02effc7072
S-BOX INPUT: 6117ba866527

*/
```

sboxoutput.cpp in gangadharashettypj/labs (master)

```
/*
lab 7
Program: SBOX output key generator
Author: Gangadhara Shetty P J
*/
#include<bits/stdc++.h>
using namespace std;

string key, previous;
int permute[]={
    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
};
int sbox[][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}
    },
    {
                {4,11,5,14,15,0,8,13,3,12,9,7,5,10,6,1},
                {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}
    }
};
void findPermutation()
{
    string key1="";
    for(int i=0;i<32;i++)
        key1 += key[permute[i]-1];
    key=key1;
}
int main()
{
    unsigned long long hexkey;
    string key1, str = "";
        cout<<"Enter a 48 bit input key in hex: ";
        cin>>hex>>hexkey;

        key=bitset<48>(hexkey).to_string();

        cout<<"Enter a 64 bit key in hex: ";
    cin>>hex>>hexkey;
        previous=bitset<64>(hexkey).to_string();

    for(int i=0, sb=0;i<48;i+=6, sb++){
        string row = "", col = "";
        row= row+key[i]+key[i+5];
        col= col+key[i+1]+key[i+2]+key[i+3]+key[i+4];
                int rowval = (int)bitset<2>(row).to_ulong();
                int colval = (int)bitset<4>(col).to_ulong();
        string tempKey = bitset<4>(sbox [sb] [rowval ] [colval] ).to_string();
                cout<<"SBOX  "<< sb + 1 <<" OUTPUT: "<<tempKey<<endl;
                key1+=tempKey;
    }

    key=key1;
        findPermutation();

        for(int i=0;i<32;i++)
                if(key[i]==previous[i]) key[i]='0';
                else key[i]='1';
        cout<<"S-BOX OUTPUT: "<<hex<< (int)bitset<32>(key).to_ulong()<<endl;
}

/*
OUTPUT
------
Enter a 48 bit input key in hex: 6117ba866527
Enter a 64 bit key in hex: cc00ccfff0aaf0aa
SBOX  1 OUTPUT: 0101
SBOX  2 OUTPUT: 1100
SBOX  3 OUTPUT: 1000
SBOX  4 OUTPUT: 0010
SBOX  5 OUTPUT: 1011
SBOX  6 OUTPUT: 0101
SBOX  7 OUTPUT: 1001
SBOX  8 OUTPUT: 0111
S-BOX OUTPUT: ef4a6544

*/
```

rc4.cpp in gangadharashettypj/labs (master)

```
/*
Program: RC4 Algorithm
Author: Gangadhara Shetty P J
*/
#include<iostream>
using namespace std;
int main()
{
    int s[256], t[256], k[256], p[256], c[256], j=0;
    string plain, cipher, key;
    cout<<"Enter plain text nad key:";
    cin>>plain>>key;

    cout<<"plaintext in bytes : " ;
        for(int i =0 ;i < plain.length() ; i++) cout<< (int)plain[i]<<" " ;
        cout<<endl;

    for(int i=0;i<255;i++)
        s[i] = i, t[i] = (int)key[i % key.length()];

    for(int i=0, j=0; i<256; i++){
        j = (j+s[i]+t[i])%256;
        int t = s[i],   s[i] = s[j],   s[j] = t;
    }

    int i=0;
    j=0;
    for(int l=0;l<plain.length();l++){
        i = (i+1)%256;
        j=(j+s[i])%256;

        int t = s[i],   s[i] = s[j],   s[j] = t;

        t= (s[i]+s[j])%256;
        k[l] = s[t];
    }
    cout<<"keystream in bytes : " ;
    for(int i =0 ;i < plain.length() ; i++) cout<< k[i]<<" " ;
    cout<<endl;

    cout<<"cipher text in bytes : " ;
    for(int i=0;i<plain.length();i++)
        c[i] = (plain[i] ^ k[i]), cout<< (int)(plain[i] ^ k[i])<< "  ";
    cout<<endl;

    cout<<"plain text in bytes after decryption : " ;
    for(int i=0;i<plain.length();i++)
        cout<<  (int)(c[i] ^ k[i])<< "  ";
    cout<<endl;
}
/*
output:
Enter plain text:CNSLAB
Enter key:BALSNC
plaintext in bytes : 67 78 83 76 65 66
keystream in bytes : 170 24 147 247 205 32
cipher text in bytes : 233   86   192   187   140   98
plain text in bytes after decryption : 67   78   83   76   65   66
*/
```

BBS and Rabin Millier.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: BBS Algorithm
Author: Gangadhara Shetty P J
*/
#include<bits/stdc++.h>
#define BIT_SIZE 16

using namespace std;

int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
bool rabinMiller(int n)
{
        int k, q=n-1;
        for(k=0; q%2==0; k++, q/=2);

    for(int i=0; i<4; i++)
    {
        int a = rand()%(n-1)+1;
        if(powModN(a,q,n) == 1)
                        return true;
        for(int j=0; j<=k-1; j++)
            if(powModN(a, pow(2,j)*q, n) == n-1)
                return true;
        }
    return false;
}
int main()
{
    long long int s, p, q, n;
    string bits="";
        srand(time(NULL));
    cout<<"Enter P, Q and Seed value: ";
    cin>>p>>q>>s;
    n = p*q;
    s=(s*s)%n;
        cout<<"bits generated: ";
    for(int i=0;i<BIT_SIZE;i++)
        s=(s*s)%n, bits+=(s%2+'0'), cout<<s%2<<" ";
    int num = bitset<BIT_SIZE>(bits).to_ulong();
    cout<<endl<<"Random Number: "<<num<<endl;
        cout<<"Rabin Miller test: ";
    if(rabinMiller(num))
                cout<<"Composite"<<endl;
        else
                cout<<"InConclusive"<<endl;

}
/*
output:
Enter P, Q and Seed value: 7 11 7
bits generated: 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
Random Number: 4369
Rabin Miller test: InConclusive
*/
```

client.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: RSA client
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
    connect(sock, (struct sockaddr *) &addr, sizeof(addr));
    return sock;
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
int gcd(int p, int q)
{
    if(q==0)    return p;
    gcd(q, p%q);
}
void itoc(int n1, int n2)
{
        string s = to_string(n1)+"|"+to_string(n2);
        strcpy(buffer, s.c_str());
}
int GetInverseDeterminant(int e ,int fi){
        for(int i=1;i<fi;i++)
                if((i*e)%fi==1) return i;
        return -1;
}
void generateKey(int p, int q, int &e, int &d, int &n)
{
        n = p*q;
    int fi=(p-1)*(q-1);
    for(int i=2;i<fi; i++)
        if(gcd(i, fi) ==1)
            {e=i;  break;}
        d = GetInverseDeterminant(e, fi);
        cout<<"Public key of server: ("<<e<<"|"<<n<<")"<<endl;
        cout<<"Private key of server: ("<<d<<"|"<<n<<")"<<endl;
}
int main()
{
    char ip[50]="127.0.0.1";
    int port=1234, p, q, e, d, n, fi, C;
    int sock = connectToServer(ip, port);

    cout << "\nEnter two prime numbers : ";
        cin >> p >> q;
    generateKey(p, q, e, d, n);

        itoc(e,n);
    send(sock, &buffer, sizeof(buffer), 0);
    cout << "\nSent Public key to server." << endl;

    recv(sock, &C, sizeof(C), 0);
    cout << "\nCiphertext received from server : " << C << endl;

    int M = powModN(C, d, n);
    cout << "\nDecrypted Text : " << M << endl << endl;
}

/*
Enter two prime numbers : 11 7
d= 43
Sent Public key to server.
Ciphertext received from server : 64
Decrypted Text : 36
*/
```

server.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: RSA server
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);
    return sock;
}
void ctoi(char buf[100], int &n1, int &n2)
{
        int i=0;
        n1=0;n2=0;
        while(buf[i]!='|')
                n1*=10, n1+=(buf[i++]-'0');
        while(buf[++i])
                n2*=10, n2+=(buf[i]-'0');
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
int main()
{
    int port=1234, e,n, M;
    int sock = createServer(port);

    recv(sock, &buffer, sizeof(buffer), 0);
        ctoi(buffer, e,n);
    cout << "\nPublic key received from client : (" << e << ", " << n << ")" << endl;

    cout << "\nEnter message(M<" << n << ") to encrypt : ";
        cin >> M;

    int C = powModN(M, e, n);
    cout << "\nEncrypted Text : " << C << endl;
    send(sock, &C, sizeof(C), 0);
    cout << "\nSent ciphertext to client." << endl << endl;
}
/*
Public key received from client : {7, 77}
Enter message(M<77) to encrypt : 36
Encrypted Text : 64
Sent ciphertext to client.
*/
```

client.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: RSA Block client
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
    connect(sock, (struct sockaddr *) &addr, sizeof(addr));
    return sock;
}

int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
int gcd(int p, int q)
{
    if(q==0)    return p;
    gcd(q, p%q);
}
void itoc(int n1, int n2)
{
        string s = to_string(n1)+"|"+to_string(n2);
        strcpy(buffer, s.c_str());
}
int GetInverseDeterminant(int e ,int fi){
        for(int i=1;i<fi;i++)
                if((i*e)%fi==1) return i;
        return -1;
}
void generateKey(int p, int q, int &e, int &d, int &n)
{
        n = p*q;
    int fi=(p-1)*(q-1);
    for(int i=2;i<fi; i++)
        if(gcd(i, fi) ==1)
            {e=i;  break;}
        d = GetInverseDeterminant(e, fi);
        cout<<"Public key of server: ("<<e<<"|"<<n<<")"<<endl;
        cout<<"Private key of server: ("<<d<<"|"<<n<<")"<<endl;
}
int main()
{
    char ip[50]="127.0.0.1";
    int port=1234;
        int p, q, e, d, n, fi, C, m;
    int sock = connectToServer(ip, port);
        string message;

    cout << "\nEnter two prime numbers : ";
        cin >> p >> q;

        generateKey(p, q, e, d, n);

        itoc(e,n);
    send(sock, &buffer, sizeof(buffer), 0);
    cout << "\nSent Public key to server." << endl;

    recv(sock, &C, sizeof(C), 0);
        cout<<"Received Encrypted Message:  ";
        for(int i=0;i<C;i++)
        {
                recv(sock, &m, sizeof(m), 0);
                message = message + (char)powModN(m, d, n);
                cout<<m<<"  ";
        }
        cout<<endl<<"Message is: "<<message<<endl;
}

/*
Enter two prime numbers : 101 131
Public key of server: (3|13231)
Private key of server: (8667|13231)
Sent Public key to server.
Received Encrypted Message:  4436  7900  12541  2767  12965  1791
Message is: cnslab
*/
```

server.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: RSA Block server
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));

    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);

    return sock;
}
void ctoi(char buf[100], int &n1, int &n2)
{
        int i=0;
        n1=0;n2=0;
        while(buf[i]!='|')
                n1*=10, n1+=(buf[i++]-'0');
        while(buf[++i])
                n2*=10, n2+=(buf[i]-'0');
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
int main()
{
    int port=1234, e,n, len;
    int sock = createServer(port);
        string message;

    recv(sock, &buffer, sizeof(buffer), 0);
        ctoi(buffer, e,n);
    cout << "\nPublic key received from client : (" << e << ", " << n << ")" << endl;

    cout << "\nEnter message to be send : ";
        cin >> message;

        len = message.length();
        send(sock, &len, sizeof(len), 0);


        cout << "\nEncrypted Message : ";
    for(int i=0; message[i]; i++)
        {
                int C = powModN(message[i], e, n);
                send(sock, &C, sizeof(C), 0);
                cout<<C<<" ";
        }
        cout<<endl<<"Message Sent"<<endl;
}
```

```
/*
Public key received from client : {3, 13231}
Enter message to be send : cnslab
Encrypted Message :   4436 7900 12541 2767 12965 1791
Message Sent
*/
```

client.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: RSA Key Exchange client
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
    connect(sock, (struct sockaddr *) &addr, sizeof(addr));
    return sock;
}
int GetInverseDeterminant(int e ,int fi){
        for(int i=1;i<fi;i++)
                if((i*e)%fi==1) return i;
        return -1;
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
void ctoi(char buf[100], int &n1, int &n2)
{
        int i=0;
        n1=0;n2=0;
        while(buf[i]!='|')
                n1*=10, n1+=(buf[i++]-'0');
        while(buf[++i])
                n2*=10, n2+=(buf[i]-'0');
}
void itoc(int n1, int n2)
{
        string s = to_string(n1)+"|"+to_string(n2);
        strcpy(buffer, s.c_str());
}
int gcd(int p, int q)
{
    if(q==0)    return p;
    gcd(q, p%q);
}
void generateKey(int p, int q, int &e, int &d, int &n)
{
        n = p*q;
    int fi=(p-1)*(q-1);
    for(int i=2;i<fi; i++)
        if(gcd(i, fi) ==1)
            {e=i;  break;}
        d = GetInverseDeterminant(e, fi);
        cout<<"Public key of client: ("<<e<<"|"<<d<<")"<<endl;
        cout<<"Private key of client: ("<<d<<"|"<<d<<")"<<endl;
}
int main()
{
    char ip[50]="127.0.0.1";
    int port=1234, pue, pre, pus, sid, nonces, noncec, cid, noncec1, ns, nc, p, q;
    int sock = connectToServer(ip, port);
        srand(time(NULL));

        cout << "\n1. Enter two prime numbers : ";
        cin >> p >> q;

        generateKey(p, q, pue, pre, nc);
        itoc(pue, nc);
        cout<<"Sending pue|n "<<buffer<<endl;
        send(sock, &buffer, sizeof(buffer), 0);

        recv(sock, &buffer, sizeof(buffer), 0);
        ctoi(buffer, pus, ns);
        cout<<"received server pus|n "<<buffer<<endl;


        recv(sock, &buffer, sizeof(buffer), 0);
        ctoi(buffer, sid,nonces);
        nonces = powModN(nonces, pre, nc);
        sid=powModN(sid, pre, nc);
        cout<<"received encrypted sid|nonces "<<buffer<<endl;
        cout<<"received decrypted sid|nonces "<<sid<<"|"<<nonces<<endl;

        noncec = rand()%100;
        itoc(powModN(nonces, pus, ns), powModN(noncec, pus, ns));
        send(sock, &buffer, sizeof(buffer), 0);
        cout<<"Sending plain nonces|noncec "<< nonces<<"|"<<noncec<<endl;
        cout<<"Sending encrypted nonces|noncec "<< buffer<<endl;

        recv(sock, &buffer, sizeof(buffer), 0);
        noncec1 = atoi(buffer);
        noncec1 = powModN(noncec1, pre, nc);
        cout<<"received encrypted noncec "<<buffer<<endl;
        cout<<"received decrypted noncec "<<noncec1<<endl;
        if(noncec!=noncec1)
                {       cout<<"Nonce din't match"<<endl;        exit(0);        }
        else    cout<<"Server Authenticated"<<endl;

        recv(sock, &buffer, sizeof(buffer), 0);
        cout<<"received encrypted key "<<buffer<<endl;
        int key = powModN(atoi(buffer), pre, nc);
        key = powModN(key, pus, ns);
        cout<<"received decrypted key "<<key<<endl;
        return 0;
}

/*
1. enter client (e|n): 7477|18281
Sending pue|n 7477|18281
2. enter client (d): 14413
received server pus|n 4551|13231
received encrypted sid|nonces 4168|11880
received decrypted sid|nonces 29|28
Sending plain nonces|noncec 28|28
Sending encrypted nonces|noncec 6840|6840
received encrypted noncec 6726
received decrypted noncec 14381
received encrypted key 5502
received decrypted key 454
*/
```

server.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: RSA Key Exchange server
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);
    return sock;
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
void ctoi(char buf[100], int &n1, int &n2)
{
        int i=0;
        n1=0;n2=0;
        while(buf[i]!='|')
                n1*=10, n1+=(buf[i++]-'0');
        while(buf[++i])
                n2*=10, n2+=(buf[i]-'0');
}
void itoc(int n1, int n2)
{
        string s = to_string(n1)+"|"+to_string(n2);
        strcpy(buffer, s.c_str());
}
void itoc(int n1)
{
        string s = to_string(n1);
        strcpy(buffer, s.c_str());
}
int GetInverseDeterminant(int e ,int fi){
        for(int i=1;i<fi;i++)
                if((i*e)%fi==1) return i;
        return -1;
}
int gcd(int p, int q)
{
    if(q==0)    return p;
    gcd(q, p%q);
}
void generateKey(int p, int q, int &e, int &d, int &n)
{
        n = p*q;
    int fi=(p-1)*(q-1);
    for(int i=2;i<fi; i++)
        if(gcd(i, fi) ==1)
            {e=i;  break;}
        d = GetInverseDeterminant(e, fi);
        cout<<"Public key of server: ("<<e<<"|"<<d<<")"<<endl;
        cout<<"Private key of server: ("<<d<<"|"<<d<<")"<<endl;
}

int main()
{
    int port=1234, sid, sid1, cid, nonces,nonces1, noncec, pue, ns, nc, pus, prs, key,
p, q;
    int sock = createServer(port);
        srand(time(NULL));

        recv(sock, &buffer, sizeof(buffer), 0);
        ctoi(buffer, pue,nc);
        cout<<"received pue|n "<<buffer<<endl;

        cout << "\n2. Enter two prime numbers : ";
        cin >> p >> q;

        generateKey(p, q, pus, prs, ns);
        itoc(pus, ns);
        cout<<"Sending pue|n "<<buffer<<endl;
        send(sock, &buffer, sizeof(buffer), 0);

        cout<<"3. Enter server ID: ";
        cin>>sid;
        nonces = rand()%100;
        itoc(powModN(sid, pue,nc), powModN(nonces, pue,nc));
        send(sock, &buffer, sizeof(buffer), 0);
        cout<<"sending plain sid|nonces "<<sid<<"|"<<nonces<<endl;
        cout<<"sending encrypted sid|nonces "<<buffer<<endl;

        recv(sock, &buffer, sizeof(buffer), 0);
        ctoi(buffer, nonces1, noncec);
        nonces1 = powModN(nonces1,prs,ns);
        noncec= powModN(noncec, prs,ns);
        cout<<"received encrypted nonces|noncec from client "<<buffer<<endl;
        cout<<"received decrypted nonces|noncec from client "<<nonces1<<"|"
<<noncec<<endl;
        if(nonces!=nonces1)
                {       cout<<"Nonce din't match"<<endl;        exit(0);        }
        else    cout<<"Client Authenticated"<<endl;

        itoc(powModN(noncec, pue, nc));
        send(sock, &buffer, sizeof(buffer), 0);
        cout<<"Sending plain noncec "<<noncec<<endl;
        cout<<"Sending encrypted noncec "<<buffer<<endl;

        cout<<"4. Enter the key: ";
        cin>>key;
        cout<<"Sending plain key "<<key<<endl;
        key = powModN(key, prs, ns);
        key = powModN(key, pue, nc);
        itoc(key);
        send(sock, &buffer, sizeof(buffer), 0);
        cout<<"Sending encrypted key "<<buffer<<endl;

}

/*
received pue|n 7477|18281
3. Enter server (e|n): 4551|13231
4. Enter server (d): 1951
5. Enter server ID: 29
sending plain sid|nonces 29|28
sending encrypted sid|nonces 4168|11880
```

```
received encrypted nonces|noncec 6840|6840
received decrypted nonces|noncec 28|28
Sending plain noncec 28
Sending encrypted noncec 28
6. Enter the key: 454
Sending plain key 454
Sending encrypted key 5502
*/
```

client.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: Diffie Hellman client
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int connectToServer(const char* ip, int port)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), inet_addr(ip)};
    connect(sock, (struct sockaddr *) &addr, sizeof(addr));
    return sock;
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
void itoa(int x)
{
        string s= to_string(x);
        strcpy(buffer, s.c_str());
}
int main()
{
    char ip[50]="127.0.0.1";
    int port=1234;
        int sock = connectToServer(ip, port);
        int q, alpha, xa, ya, yb, cipher, key, message;

        cout<<"1. Enter prime number  and primitive root: ";
    cin>>q>>alpha;

        cout<<"2. Enter private key of client (<"<<q<<") : ";
    cin>>xa;

        ya =powModN(alpha, xa, q);

        recv(sock, &buffer, sizeof(buffer), 0);
        yb =  atoi(buffer);

        itoa(ya);
        send(sock, &buffer, sizeof(buffer), 0);

        cout<<"public key of client = "<<ya<<endl;
        cout<<"received public key of server is : "<<yb<<endl;

    key = powModN(yb, xa,q);
    cout <<"secret key of client = "<< key<<endl;

        cout<<"Enter a message: ";
    cin>>message;
    cipher = message ^ key;
        cout<<"Encrypted message :"<<cipher<<endl;
        itoa(cipher);
        send(sock, &buffer, sizeof(buffer), 0);

        recv(sock, &buffer, sizeof(buffer), 0);
        cipher = atoi(buffer);
        message = cipher ^ key;
        cout<<"received encrypted message is: "<<cipher<<endl;
        cout<<"received decrypted message is: "<<(cipher^key)<<endl;

        return 0;
}

/*
1. Enter prime number  and primitive root: 761  6
2. Enter private key of client (<761) : 100
public key of client = 399
received public key of server is : 152
secret key of client = 357
Enter a message: 76
Encrypted message :297
received encrypted message is: 325
received decrypted message is: 32
*/
```

server.cpp in gangadharashettypj/labs (master)

```cpp
/*
Program: Diffie Hellman server
Author: Gangadhara Shetty P J
*/
# include <bits/stdc++.h>
# include <arpa/inet.h>
using namespace std;
char buffer[100];
int createServer(int port)
{
    int sersock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in addr = {AF_INET, htons(port), INADDR_ANY};
    bind(sersock, (struct sockaddr *) &addr, sizeof(addr));
    listen(sersock, 5);
    int sock = accept(sersock, NULL, NULL);
    return sock;
}
int powModN(int num,int p,int n)
{
        int res=1;
        for(int i=0; i<p; i++)
        res = (res * num) % n;
        return res;
}
void itoa(int x)
{
        string s= to_string(x);
        strcpy(buffer, s.c_str());
}
int main()
{
    int port=1234, q, alpha, xb, ya, yb, cipher, key, message;
        int sock = createServer(port);

        cout<<"3. Enter prime number  and primitive root: ";
    cin>>q>>alpha;

        cout<<"4. Enter private key server : ";
    cin>>xb;

        yb = powModN(alpha, xb,q);
        itoa(yb);
        send(sock, &buffer, sizeof(buffer), 0);

        recv(sock, &buffer, sizeof(buffer), 0);
        ya = atoi(buffer);

        cout<<"public key of client = "<<ya<<endl;
        cout<<"public key of server is : "<<yb<<endl;

    key = powModN(ya, xb,q);
    cout <<"secret key of server = "<< key<<endl;

        recv(sock, &buffer, sizeof(buffer), 0);
        message =atoi(buffer);
        cipher = message^key;
        cout<<"received encrypted message is: "<<message<<endl;
        cout<<"received decrypted message is: "<<cipher<<endl;

    cout<<"Enter a message: ";
    cin>>message;
    cipher = message ^ key;
        cout<<"Encrypted message :"<<cipher<<endl;
        itoa(cipher);
        send(sock, &buffer, sizeof(buffer), 0);
        return 0;
}

/*
3. Enter prime number  and primitive root: 761  6
4. Enter private key server : 200
public key of client = 399
public key of server is : 152
secret key of server = 357
received encrypted message is: 297
received decrypted message is: 76
Enter a message: 32
Encrypted message :325
*/
```