### EXPERIMENT-1

**Aim:-**Write program forCeaser cipher Mono alphabetic cipher.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

**1.Ceaser cipher**

#include <iostream>

#include <string>

#include <cctype>

using namespace std;

string caesarCipher(const string &message, int shift)

{

    string encryptedMessage = "";

    for (char ch : message) {

        if (isalpha(ch)) {

            char base = (isupper(ch)) ? 'A' : 'a';

            char encryptedChar = (ch - base + shift) % 26 + base;

            encryptedMessage += encryptedChar;

        } else {

            encryptedMessage += ch;

        }

    }

    return encryptedMessage;

}

int main() {

    string message;

    int shift;

    cout<< "Enter the message: ";

    getline(cin, message);

    cout<< "Enter the shift value: ";

    cin>> shift;

    string encryptedMessage = caesarCipher(message, shift);

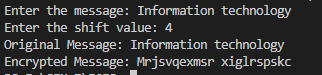
    cout<< "Original Message: " << message <<endl;

    cout<< "Encrypted Message: " <<encryptedMessage<<endl;

    return 0;

}

**Output:-**



**2.Monoalphabetic**

#include<iostream>

#include<string>

#include<vector>

using namespace std;

int main(){

string s;

cout<<"Enter plain text :";

getline(cin,s);

cout<<"Enter key :";

string key;

getline(cin,key);

vector<int>pt(26,-1);

vector<int>ky(26,-1);

cout<<"Encrypted message is : ";

int j=0;

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

if(s[i]==' '){

cout<<s[i];

continue;

}

if(s[i]>='a' && s[i]<='z'){

if(pt[s[i]-'a']!=-1){

cout<<(char)(pt[s[i]-'a']);

continue;

}

while(key[j]==' '){

j++;

if(j==key.length()){

j=0;

for(int l=0;l<26;l++) ky[l]=-1;

}

}

while(j<key.length() &&ky[key[j]-'a']!=-1){

j++;

if(j==key.length()){

j=0;

for(int l=0;l<26;l++) ky[l]=-1;

}

}

ky[key[j]-'a']=1;

pt[s[i]-'a']=key[j];

cout<<key[j];

j++;

}

}

cout<<endl;

return 0;

}

**Output:**



### EXPERIMENT-2

**Aim:-**Implementation of Play Fair cipher.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

#include <bits/stdc++.h>

using namespace std;

int main()

{

string pt, key;

cout<< "Enter plain text : ";

getline(cin, pt);

cout<< "Enter key : ";

getline(cin, key);

// matrix creation

vector<int>keyCheck(26, 0);

vector<pair<int, int>>indexMat(26);

vector<vector<char>>mat(5, vector<char>(5));

int a = 0, b = 0;

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

{

if (key[i] == 'i' || key[i] == 'j')

{

if (key[i] == 'i' &&keyCheck[key[i] - 'a'] != 1)

{

mat[a][b] = key[i];

indexMat[key[i] - 'a'] = {a, b};

b++;

if (b == 5)

{

b = 0;

a++;

}

keyCheck['i' - 'a'] = 1;

keyCheck['j' - 'a'] = 1;

}

else if (key[i] == 'j' &&keyCheck[key[i] - 'a'] != 1)

{

mat[a][b] = 'i';

indexMat[key[i] - 'a'] = {a, b};

b++;

if (b == 5)

{

b = 0;

a++;

}

keyCheck['i' - 'a'] = 1;

keyCheck['j' - 'a'] = 1;

}

}

else

{

if (key[i] >= 'a' && key[i] <= 'z' &&keyCheck[key[i] - 'a'] != 1)

{

mat[a][b] = key[i];

indexMat[key[i] - 'a'] = {a, b};

b++;

if (b == 5)

{

b = 0;

a++;

}

keyCheck[key[i] - 'a'] = 1;

}

}

}

for (int i = 0; i< 26; i++)

{

if (keyCheck[i] != 1)

{

if (i == 8)

{

mat[a][b] = 'i';

indexMat[i] = {a, b};

b++;

if (b == 5)

{

b = 0;

a++;

}

keyCheck[i] = 1;

keyCheck[i + 1] = 1;

}

mat[a][b] = (char)(i + 'a');

indexMat[i] = {a, b};

b++;

if (b == 5)

{

b = 0;

a++;

}

keyCheck[i] = 1;

}

}

cout<< "\nMatrixis : \n"

<<endl;

for (int i = 0; i< 5; i++)

{

for (int j = 0; j < 5; j++)

{

cout<< mat[i][j] << " ";

}

cout<<endl;

}

// encryption

string ans;

char a1, a2;

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

{

a1 = '$';

a2 = '$';

while (pt[i] == ' ' &&i<pt.length())

{

i++;

}

if (i>= pt.length())

{

break;

}

a1 = pt[i];

i++;

while (pt[i] == ' ' &&i<pt.length())

{

i++;

}

if (i == pt.length())

{

a2 = 'x';

}

else if (pt[i] == a1)

{

a2 = 'x';

i--;

}

else

{

a2 = pt[i];

}

auto t1 = indexMat[a1 - 'a'];

auto t2 = indexMat[a2 - 'a'];

if (t1.first == t2.first)

{

ans.push\_back(mat[t1.first][(t1.second + 1) % 5]);

ans.push\_back(mat[t2.first][(t2.second + 1) % 5]);

}

else if (t1.second == t2.second)

{

ans.push\_back(mat[(t1.first + 1) % 5][t1.second]);

ans.push\_back(mat[(t2.first + 1) % 5][t2.second]);

}

else

{

ans.push\_back(mat[t1.first][t2.second]);

ans.push\_back(mat[t2.first][t1.second]);

}

}

cout<< "\nYour encryption is : ";

cout<<ans;

string decans;

cout<< "\nYour decryption is : ";

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

{

a1 = '$';

a2 = '$';

while (ans[i] == ' ' &&i<ans.length())

{

i++;

}

if (i>= ans.length())

break;

a1 = ans[i];

i++;

while (ans[i] == ' ' &&i<ans.length())

{

i++;

}

if (i == ans.length())

a2 = 'x';

else if (ans[i] == a1)

{

a2 = 'x';

i--;

}

else

a2 = ans[i];

auto t1 = indexMat[a1 - 'a'];

auto t2 = indexMat[a2 - 'a'];

if (t1.first == t2.first)

{

decans.push\_back(mat[t1.first][((t1.second - 1)==-1)?4:(t1.second - 1)]);

decans.push\_back(mat[t2.first][((t2.second - 1)==-1)?4:(t2.second - 1)]);

}

else if (t1.second == t2.second)

{

decans.push\_back(mat[((t1.first - 1)==-1)?4:(t1.first-1)][t1.second]);

decans.push\_back(mat[((t2.first - 1)==-1)?4:(t2.first-1)][t2.second]);

}

else

{

decans.push\_back(mat[t1.first][t2.second]);

decans.push\_back(mat[t2.first][t1.second]);

}

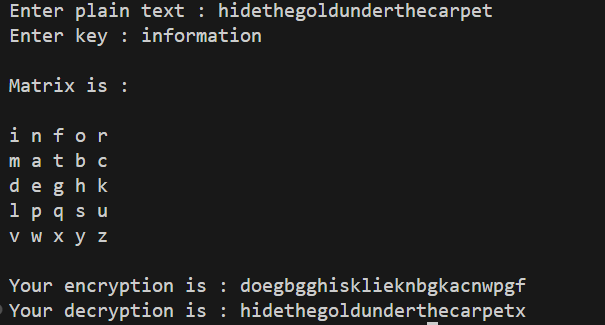
}

cout<<decans;

return 0;

}

**Output:**

****

### EXPERIMENT-3

**Aim:-**Write program forHill cipher.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

#include <bits/stdc++.h>

using namespace std;

int determinantOfMatrix(int mat[2][2], int n)

{

int num1, num2, det = 1, index, total = 1; // Initialize result

// temporary array for storing row

int temp[n + 1];

// loop for traversing the diagonal elements

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

{

index = i; // initialize the index

// finding the index which has non zero value

while (index < n && mat[index][i] == 0)

{

index++;

}

if (index == n) // if there is non zero element

{

// the determinant of matrix as zero

continue;

}

if (index != i)

{

// loop for swapping the diagonal element row and

// index row

for (int j = 0; j < n; j++)

{

swap(mat[index][j], mat[i][j]);

}

// determinant sign changes when we shift rows

// go through determinant properties

det = det \* pow(-1, index - i);

}

// storing the values of diagonal row elements

for (int j = 0; j < n; j++)

{

temp[j] = mat[i][j];

}

// traversing every row below the diagonal element

for (int j = i + 1; j < n; j++)

{

num1 = temp[i]; // value of diagonal element

num2 = mat[j][i]; // value of next row element

// traversing every column of row

// and multiplying to every row

for (int k = 0; k < n; k++)

{

// multiplying to make the diagonal

// element and next row element equal

mat[j][k] = (num1 \* mat[j][k]) - (num2 \* temp[k]);

}

total = total \* num1; // Det(kA)=kDet(A);

}

}

// multiplying the diagonal elements to get determinant

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

{

det = det \* mat[i][i];

}

return (det / total); // Det(kA)/k=Det(A);

}

int main()

{

string pt;

string key;

cout<< "Enter plain text : ";

getline(cin, pt);

cout<< "Enter key text(4 character) : ";

getline(cin, key);

string s;

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

{

if (pt[i] >= 'a' &&pt[i] <= 'z')

{

s.push\_back(pt[i]);

}

}

string ct;

for (int i = 0; i<s.length(); i += 2)

{

if (i + 1 == s.length())

{

s.push\_back('x');

}

if (s[i] >= 'a' && s[i] <= 'z' &&s[i + 1] >= 'a' && s[i + 1] <= 'z')

{

int ans = ((s[i] - 'a') \* (key[0] - 'a')) + ((s[i + 1] - 'a') \* (key[2] - 'a'));

ans = ans % 26;

ct.push\_back((char)(ans + 'a'));

ans = ((s[i] - 'a') \* (key[1] - 'a')) + ((s[i + 1] - 'a') \* (key[3] - 'a'));

ans = ans % 26;

ct.push\_back((char)(ans + 'a'));

}

}

cout<< "\nYour Encryption is : " <<ct<< "\n"

<<endl;

// for (int i = 0; i<ct.length(); i += 2)

// {

// cout<< (ct[i] - 'a') << " " << (ct[i + 1] - 'a') <<endl;

// }clear

int q, a, b, r, t1, t2, t;

a = 26;

int mat[2][2] = {{(key[0] - 'a'), (key[1] - 'a')}, {(key[2] - 'a'), (key[3] - 'a')}};

b = determinantOfMatrix(mat, 2);

t1 = 0;

t2 = 1;

do

{

q = a / b;

r = a % b;

t = t1 - (t2 \* q);

a = b;

b = r;

t1 = t2;

t2 = t;

} while (r != 0);

// cout<< t1 <<endl;

s = ct;

string pta;

for (int i = 0; i<s.length(); i += 2)

{

if (s[i] >= 'a' && s[i] <= 'z' &&s[i + 1] >= 'a' && s[i + 1] <= 'z')

{

int ans = ((s[i] - 'a') \* (key[3] - 'a')) + ((s[i + 1] - 'a') \* (-1) \* (key[2] - 'a'));

ans \*= t1;

ans = ans % 26;

if (ans< 0)

{

ans = 26 + ans;

}

// cout<<ans<< " ";

pta.push\_back((char)(ans + 'a'));

ans = ((s[i] - 'a') \* (-1) \* (key[1] - 'a')) + ((s[i + 1] - 'a') \* (key[0] - 'a'));

ans \*= t1;

ans = ans % 26;

if (ans< 0)

{

ans = 26 + ans;

}

// cout<<ans<< " ";

pta.push\_back((char)(ans + 'a'));

}

}

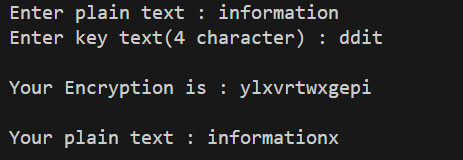
cout<< "\nYour plain text : " <<pta<< "\n"

<<endl;

return 0;

}

**Output:-**

****

### EXPERIMENT-4

**Aim:-**S-DES algorithm for data encryption along with key generation of S-DES.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

#include <bits/stdc++.h>

using namespace std;

int findRowColS1(int s1[4][4], string a)

{

int i, j;

if (a[1] == '0' && a[2] == '0')

{

i = 0;

}

else if (a[1] == '1' && a[2] == '0')

{

i = 2;

}

else if (a[1] == '0' && a[2] == '1')

{

i = 1;

}

else

{

i = 3;

}

if (a[0] == '0' && a[3] == '0')

{

j = 0;

}

else if (a[0] == '1' && a[3] == '0')

{

j = 2;

}

else if (a[0] == '0' && a[3] == '1')

{

j = 1;

}

else

{

j = 3;

}

return s1[j][i];

}

int findRowColS0(int s0[4][4], string a)

{

int i, j;

if (a[1] == '0' && a[2] == '0')

{

i = 0;

}

else if (a[1] == '1' && a[2] == '0')

{

i = 2;

}

else if (a[1] == '0' && a[2] == '1')

{

i = 1;

}

else

{

i = 3;

}

if (a[0] == '0' && a[3] == '0')

{

j = 0;

}

else if (a[0] == '1' && a[3] == '0')

{

j = 2;

}

else if (a[0] == '0' && a[3] == '1')

{

j = 1;

}

else

{

j = 3;

}

return s0[j][i];

}

string xorString(string a, string b, int n)

{

string ans;

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

{

if ((a[i] == '0' && b[i] == '0') || (a[i] == '1' && b[i] == '1'))

{

ans.push\_back('0');

}

else

{

ans.push\_back('1');

}

}

return ans;

}

int main()

{

// 00001011

string pt, keyTemp;

cout<< "Enter plain text : ";

cin>>pt;

cout<< "Enter key : ";

cin>>keyTemp;

string key;

vector<int> p10(10);

cout<< "Enter p10 (in 10 number): ";

for (int i = 0; i< 10; i++)

{

int x;

cin>> x;

p10[i] = x;

key.push\_back(keyTemp[x - 1]);

}

// 10 bit key partition

string fbit = key.substr(0, 5);

string sbit = key.substr(5, 5);

// 1 shifting

string lcs1fbit, lcs1sbit;

for (int i = 0; i< 5; i++)

{

lcs1fbit.push\_back(fbit[(i + 1) % 5]);

lcs1sbit.push\_back(sbit[(i + 1) % 5]);

}

string lcs1 = lcs1fbit + lcs1sbit;

// 2 shifting

string lcs2fbit, lcs2sbit;

for (int i = 0; i< 5; i++)

{

lcs2fbit.push\_back(lcs1fbit[(i + 2) % 5]);

lcs2sbit.push\_back(lcs1sbit[(i + 2) % 5]);

}

string lcs2 = lcs2fbit + lcs2sbit;

// calculating k1,k2

string k1, k2;

vector<int> p8(8);

cout<< "Enter P8 (in 8 number): ";

for (int i = 0; i< 8; i++)

{

int x;

cin>> x;

p8[i] = x;

k1.push\_back(lcs1[x - 1]);

k2.push\_back(lcs2[x - 1]);

}

cout<< "k1 : " << k1 <<endl;

cout<< "k2 : " << k2 <<endl;

// Encryption

vector<int>ip(8);

string ptip;

cout<< "Enter IP (in 8 number) : ";

for (int i = 0; i< 8; i++)

{

int x;

cin>> x;

ip[i] = x;

ptip.push\_back(pt[x - 1]);

}

vector<int>exp(8);

string rightExp;

cout<< "Enter expanded permutation (in 8 number) : ";

for (int i = 0; i< 8; i++)

{

int x;

cin>> x;

exp[i] = x;

rightExp.push\_back(ptip[x - 1 + 4]);

}

// rightExp XOR k1

string afterf1 = xorString(rightExp, k1, 8);

int s0[4][4] = {{1, 0, 3, 2}, {3, 2, 1, 0}, {0, 2, 1, 3}, {3, 1, 3, 2}};

int s1[4][4] = {{0, 1, 2, 3}, {2, 0, 1, 3}, {3, 0, 1, 0}, {2, 1, 0, 3}};

int a, b;

a = findRowColS0(s0, afterf1.substr(0, 4));

b = findRowColS1(s1, afterf1.substr(4, 4));

cout<< "a b : " << a << " " << b <<endl;

string afterS0S1;

if (a == 0)

{

afterS0S1.push\_back('0');

afterS0S1.push\_back('0');

}

else if (a == 1)

{

afterS0S1.push\_back('0');

afterS0S1.push\_back('1');

}

else if (a == 2)

{

afterS0S1.push\_back('1');

afterS0S1.push\_back('0');

}

else

{

afterS0S1.push\_back('1');

afterS0S1.push\_back('1');

}

if (b == 0)

{

afterS0S1.push\_back('0');

afterS0S1.push\_back('0');

}

else if (b == 1)

{

afterS0S1.push\_back('0');

afterS0S1.push\_back('1');

}

else if (b == 2)

{

afterS0S1.push\_back('1');

afterS0S1.push\_back('0');

}

else

{

afterS0S1.push\_back('1');

afterS0S1.push\_back('1');

}

cout<< "afterS0S1 : " << afterS0S1 <<endl;

cout<< "Enter p4 (in 4 number) : ";

vector<int> p4(4);

string rightWithoutXor;

for (int i = 0; i< 4; i++)

{

int x;

cin>> x;

p4[i] = x;

rightWithoutXor.push\_back(afterS0S1[x - 1]);

}

string finalRight = xorString(rightWithoutXor, ptip.substr(0, 4), 4);

cout<< "finalRight : " <<finalRight<<endl;

// swap done

string afterFun1 = ptip.substr(4, 4) + finalRight;

cout<< "afterFun1 : " << afterFun1 <<endl;

// now ptip is afterFun1

ptip = afterFun1;

string rightExp2;

for (int i = 0; i< 8; i++)

{

rightExp2.push\_back(ptip[exp[i] - 1 + 4]);

}

cout<< "rightExp2 : " << rightExp2 <<endl;

string afterf2 = xorString(rightExp2, k2, 8);

cout<< "afterf2 : " << afterf2 <<endl;

a = findRowColS0(s0, afterf2.substr(0, 4));

b = findRowColS1(s1, afterf2.substr(4, 4));

string afterS0S12nd;

if (a == 0)

{

afterS0S12nd.push\_back('0');

afterS0S12nd.push\_back('0');

}

else if (a == 1)

{

afterS0S12nd.push\_back('0');

afterS0S12nd.push\_back('1');

}

else if (a == 2)

{

afterS0S12nd.push\_back('1');

afterS0S12nd.push\_back('0');

}

else

{

afterS0S12nd.push\_back('1');

afterS0S12nd.push\_back('1');

}

if (b == 0)

{

afterS0S12nd.push\_back('0');

afterS0S12nd.push\_back('0');

}

else if (b == 1)

{

afterS0S12nd.push\_back('0');

afterS0S12nd.push\_back('1');

}

else if (b == 2)

{

afterS0S12nd.push\_back('1');

afterS0S12nd.push\_back('0');

}

else

{

afterS0S12nd.push\_back('1');

afterS0S12nd.push\_back('1');

}

cout<< "afterS0S12nd : " << afterS0S12nd <<endl;

string rightWithoutXor2nd;

for (int i = 0; i< 4; i++)

{

rightWithoutXor2nd.push\_back(afterS0S12nd[p4[i] - 1]);

}

cout<< "rightWithoutXor2nd : " << rightWithoutXor2nd <<endl;

string finalRight2nd = xorString(rightWithoutXor2nd, ptip.substr(0, 4), 4);

cout<< "finalRight2nd : " << finalRight2nd <<endl;

string afterFun2 = finalRight2nd + ptip.substr(4, 4);

cout<< "afterFun2 : " << afterFun2 <<endl;

string ans;

cout<< "Enter IP^(-1) : ";

vector<int> ip1(8);

for (int i = 0; i< 8; i++)

{

int x;

cin>> x;

ip1[i] = x;

ans.push\_back(afterFun2[x - 1]);

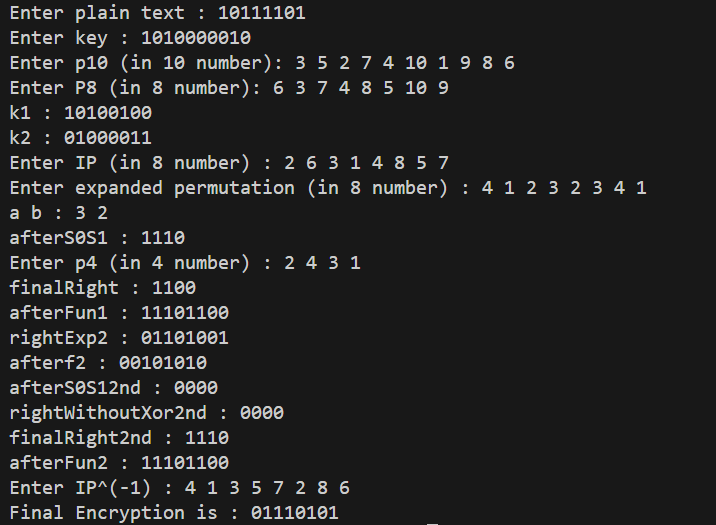
}

cout<< "Final Encryption is : " <<ans<<endl;

return 0;

}

**Output:-**

****

### EXPERIMENT-5

**Aim:-**Write a program to generate and exchange public keys using client server mechanism

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Server.java**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class Server {

public static booleanisPrime(int n) {

if (n <= 1) {

return false;

}

if (n <= 3) {

return true;

}

if (n % 2 == 0 || n % 3 == 0) {

return false;

}

for (int i = 5; i \* i<= n; i += 6) {

if (n % i == 0 || n % (i + 2) == 0) {

return false;

}

}

return true;

}

public static void main(String[] args) {

try {

ServerSocketserverSocket = new ServerSocket(12345);

System.out.println("Waiting for client...");

Socket clientSocket = serverSocket.accept();

System.out.println("Client connected!");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter prime number p: ");

int p = scanner.nextInt();

System.out.print("Enter prime number q: ");

int q = scanner.nextInt();

System.out.print("Enter prime number e: ");

int e=scanner.nextInt();

if(isPrime(p) &&isPrime(q)){

int n = p \* q;

int phi\_n = (p - 1) \* (q - 1);

int d = calculateModInverse(e, phi\_n);

System.out.println("Value of d is:"+d);

DataOutputStream out = new DataOutputStream(clientSocket.getOutputStream());

out.writeInt(n);

out.writeInt(e);

out.writeInt(d);

clientSocket.close();

serverSocket.close();

}else{

System.out.println("Check whether p and q are ");

} }

catch (Exception e) {

e.printStackTrace();

}

}

private static int calculateModInverse(int a, int m) {

a = a % m;

for (int x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return 1;

}

}

**Client.cpp:**

import java.io.\*;

import java.net.\*;

import java.math.BigInteger;

import java.util.Scanner;

public class Client {

public static void main(String[] args) {

try {

Socket socket = new Socket("127.0.0.1", 12345);

DataInputStream in = new DataInputStream(socket.getInputStream());

int n = in.readInt();

int e = in.readInt();

int d=in.readInt();

System.out.println("Public Key{e,n}: {" + e + ", " + n + "}");

System.out.println("Public Key{d,n}: {" + d + ", " + n + "}");

// Use the received public key for encryption

Scanner scanner = new Scanner(System.in);

System.out.print("Enter message to encrypt: ");

double message = scanner.nextDouble();

double encryptedMessage = encrypt(message, e, n);

System.out.println("Encrypted Message: " + encryptedMessage);

double decryptedMessage=decrypt(encryptedMessage,d,n);

System.out.println("Decrypted Message: " + decryptedMessage);

socket.close();

} catch (Exception e) {

e.printStackTrace();

}

}

private static double encrypt(double message, int e, int n) {

double encrypted=modPow(message,e,n);

return encrypted;

}

public static double modPow(double base, double exponent, double modulus) {

double result = 1.0;

base = base % modulus;

while (exponent > 0) {

if (exponent % 2 == 1) {

result = (result \* base) % modulus;

}

exponent = Math.floor(exponent / 2); // Use Math.floor to ensure integer division

base = (base \* base) % modulus;

}

return result;

}

private static double decrypt(double message, int d, int n) {

double decrypted=modPow(message,d,n);

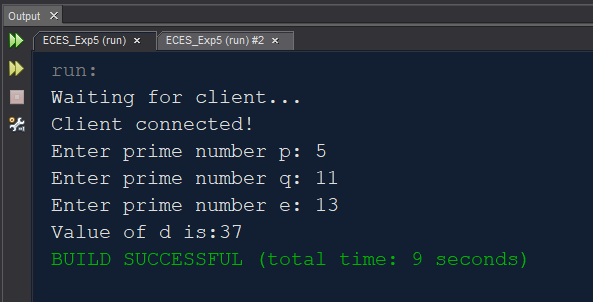
return decrypted;

}

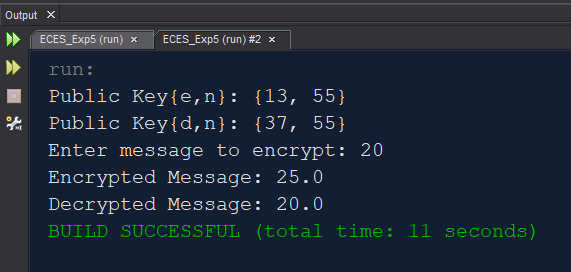
}

**Output:**

**Server:**

****

**Client:**

****

### EXPERIMENT-6

**Aim:-**Perform Encryption, Authentication and both using RSA. (Use public key shared in

above practical).

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

/\* Encrypt pain text "HI" using rsaalgoritham for the given data p=53,q=59 \*/

#include <bits/stdc++.h>

using namespace std;

int power(int x, int y, int p)

{

int res = 1; // Initialize result

x = x % p; // Update x if it is more than orequal to p

if (x == 0)

return 0; // In case x is divisible by p;

while (y > 0)

{

// If y is odd, multiply x with result

if (y & 1)

res = (res \* x) % p;

// y must be even now

y = y >> 1; // y = y/2

x = (x \* x) % p;

}

return res;

}

int main()

{

int m = 0;

string pt;

cout<< "Enter plain text : ";

cin>>pt;

string ptm;

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

{

ptm += to\_string((int)(pt[i] - 'a'));

}

m = stoi(ptm);

cout<< "your plain text in number is : " << m <<endl;

int p, q, d;

cout<< "Enter value of p : ";

cin>> p;

cout<< "Enter value of q : ";

cin>> q;

int n, fn, e = 2;

n = p \* q;

fn = (p - 1) \* (q - 1);

// gcd(fn,e)=17

while (true)

{

if (\_\_gcd(fn, e) == 1)

{

break;

}

e++;

}

cout<< "e is : " << e <<endl;

// e\*d = 1 mod fn

// find using inverse module function

int t1 = 0;

int t2 = 1;

int a = fn;

int b = e;

int r, qo, t;

do

{

qo = a / b;

r = a % b;

t = t1 - (t2 \* qo);

a = b;

b = r;

t1 = t2;

t2 = t;

} while (r != 0);

d = t1;

if (d < 0)

{

d = fn + d;

}

cout<< "d is : " << d <<endl;

cout<< "Public key : { " << e << " , " << n << " }" <<endl;

cout<< "Private key : { " << d << " , " << n << " }" <<endl;

// Encryption

// c=m^e mod n

int c = power(m, e, n);

string cstr;

string cipher;

cstr = to\_string(c);

if (cstr.length() == 2)

{

cipher.push\_back((char)(cstr[0] - '0' + 'a'));

cipher.push\_back((char)(cstr[1] - '0' + 'a'));

cout<< "Cipher text is : " << cipher <<endl;

cout<< "Cipher text is : " << c <<endl;

}

else if (cstr.length() == 3)

{

cout<< "Can't convert into text." <<endl;

cout<< "Cipher text is : " << c <<endl;

}

else

{

int x, y;

x = ((int)(cstr[0] - '0') \* 10) + (int)(cstr[1] - '0');

y = ((int)(cstr[2] - '0') \* 10) + (int)(cstr[3] - '0');

x = x % 26;

y = y % 26;

cipher.push\_back((char)(x + 'a'));

cipher.push\_back((char)(y + 'a'));

cout<< "Cipher text is : " << cipher <<endl;

cout<< "Cipher text is : " << c <<endl;

}

// Decryption

// m=c^d mod n

int m1 = power(c, d, n);

string plain;

cstr = to\_string(m1);

if (cstr.length() == 2)

{

plain.push\_back((char)(cstr[0] - '0' + 'a'));

plain.push\_back((char)(cstr[1] - '0' + 'a'));

cout<< "Plain text is : " << plain <<endl;

cout<< "Plain text is : " << m1 <<endl;

}

else if (cstr.length() == 3)

{

cout<< "Can't convert into text." <<endl;

cout<< "Plain text is : " << m1 <<endl;

}

else

{

int x, y;

x = ((int)(cstr[0] - '0') \* 10) + (int)(cstr[1] - '0');

y = ((int)(cstr[2] - '0') \* 10) + (int)(cstr[3] - '0');

x = x % 26;

y = y % 26;

plain.push\_back((char)(x + 'a'));

plain.push\_back((char)(y + 'a'));

cout<< "Plain text is : " << plain <<endl;

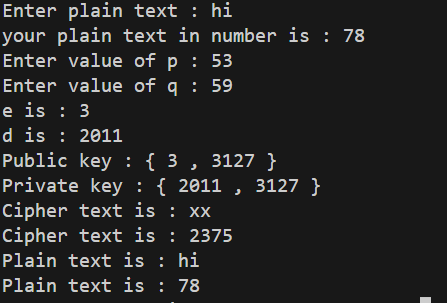
cout<< "Plain text is : " << m1 <<endl;

}

return 0;

}

**Output:-**

****

### EXPERIMENT-7

**Aim:-**Write a program to implement Diffie-Hellman Key exchange algorithm and perform

encryption and decryption.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Program :**

#include <bits/stdc++.h>

using namespace std;

int power(int x, int y, int p)

{

int res = 1;

x = x % p;

if (x == 0)

return 0;

while (y > 0)

{

if (y & 1)

res = (res \* x) % p;

y = y >> 1;

x = (x \* x) % p;

}

return res;

}

int main()

{

int alpha, p, xa, xb, ya, yb, ka, kb;

cout<< "Enter p : ";

cin>> p;

cout<< "Enter alpha : ";

cin>> alpha;

cout<< "Enter private key of user A : ";

cin>>xa;

cout<< "Enter private key of user B : ";

cin>>xb;

ya = power(alpha, xa, p);

yb = power(alpha, xb, p);

ka = power(yb, xa, p);

kb = power(ya, xb, p);

cout<< "YA : " <<ya<<endl;

cout<< "YB : " <<yb<<endl;

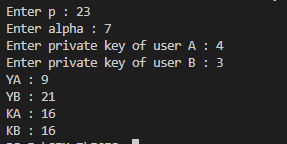
cout<< "KA : " << ka <<endl;

cout<< "KB : " << kb <<endl;

return 0;

}

**Output :**



### EXPERIMENT-8

**Aim:-**Write a program to authenticate a user with system using MD5 or SHA-1 Hashing

Technique.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.\*;

public class Md5 {

public static String generateMd5(String s) {

try {

MessageDigest md = MessageDigest.getInstance("MD5");

byte[] bit = md.digest(s.getBytes());

BigInteger bi = new BigInteger(1, bit);

String hashValue = bi.toString(16);

while (hashValue.length() < 32) {

hashValue = "0" + hashValue;

}

return hashValue;

} catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

public static void main(String args[]) throws NoSuchAlgorithmException {

Scanner sc = new Scanner(System.in);

System.out.print("Enter Message : ");

String str = sc.nextLine();

System.out.println("Your HashCode Generated by MD5 is : " + generateMd5(str));

sc.close();

}

}

**Output:**



### EXPERIMENT-9

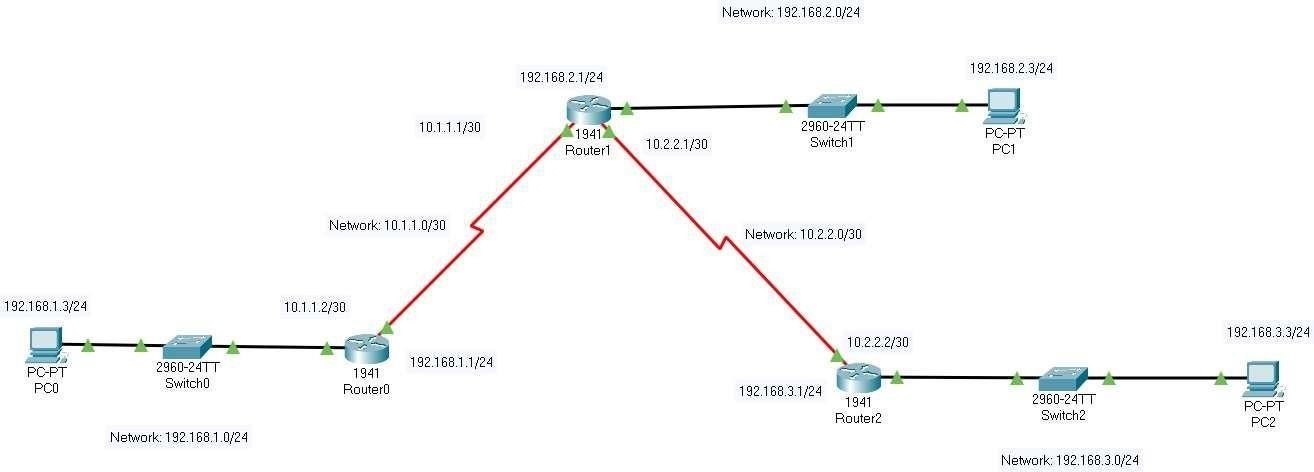
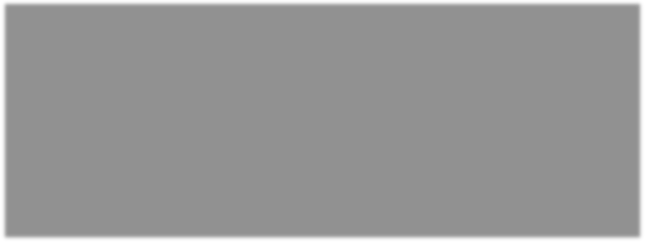
**Aim:-**Configure VPN using Packet Tracer and demonstrate the importance of IPSec**.**

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

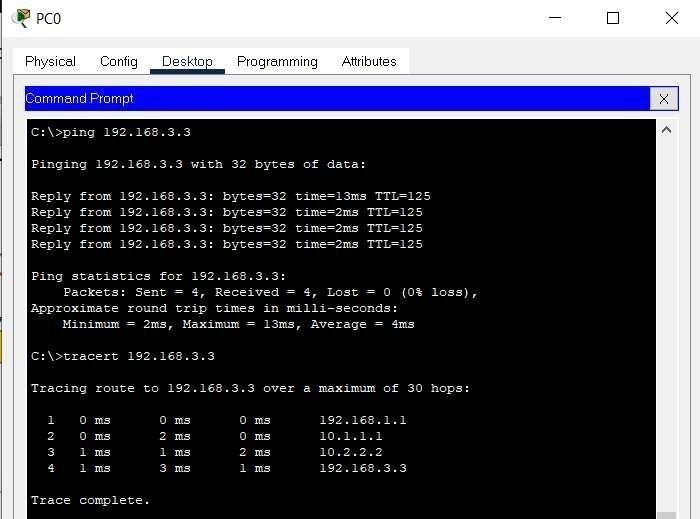
Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

**Topologyfortheconfiguration**



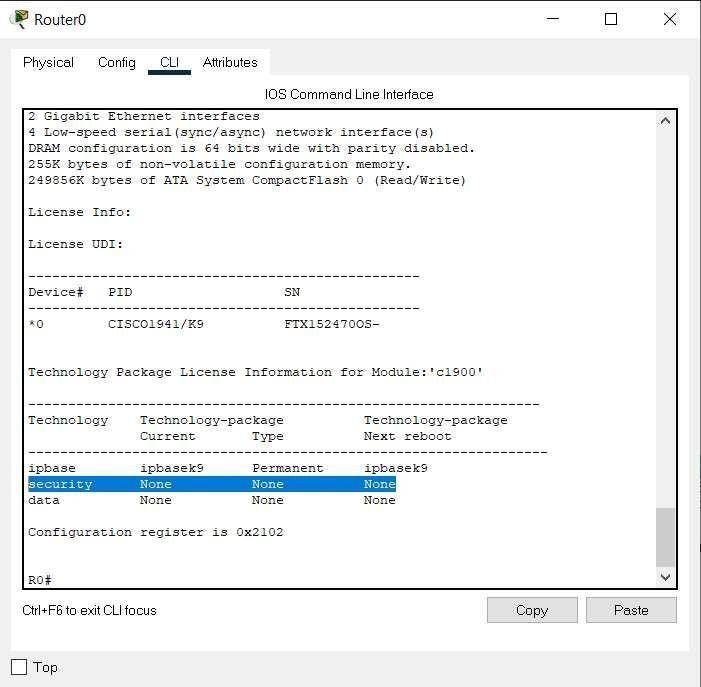
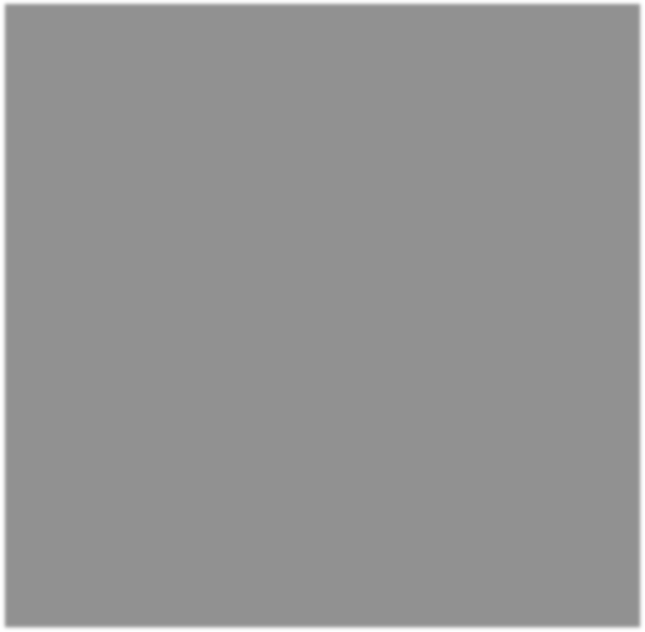
**Connectionbetween2endpoints**



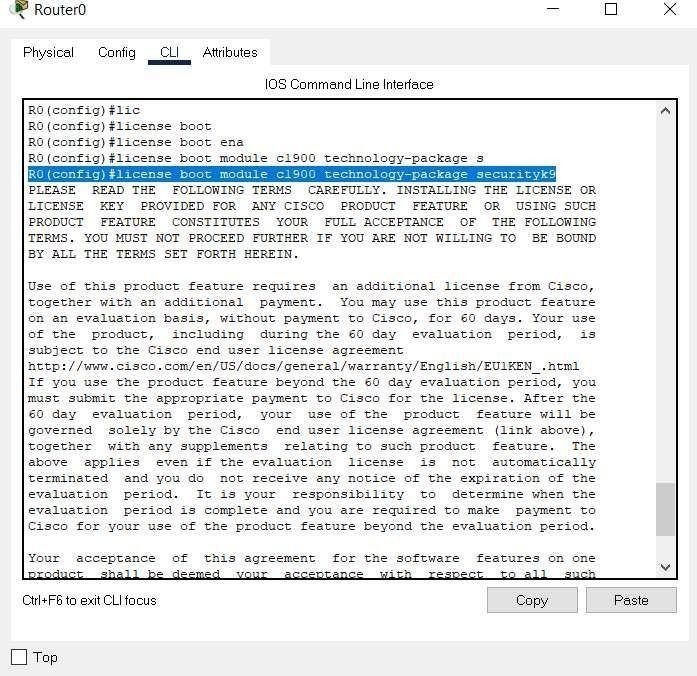
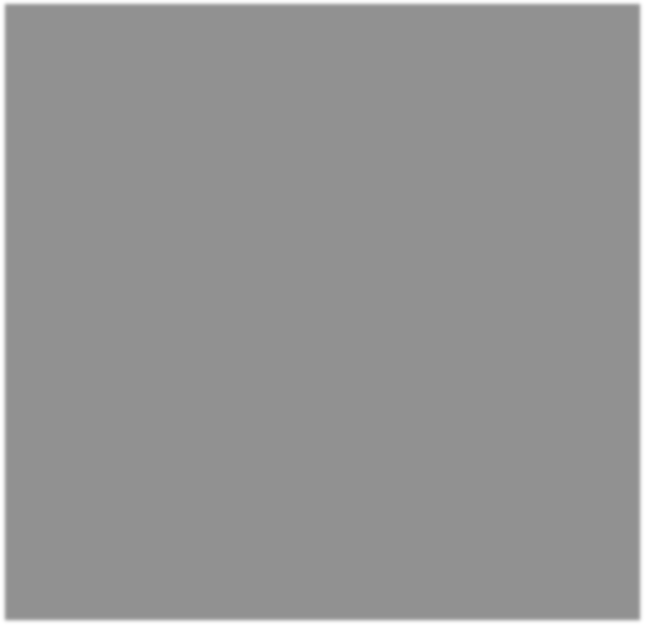
**Part1:Enablesecurityfeatures**

1. IssuetheshowversioncommandintheuserEXECorprivilegedEXECmodetoverifythat thesecuritytechnology package licenseisactivated.
2. If not, activate the securityk9 module for the next boot of the router, accept thelicense, save the configuration, and reboot R1(config)# license boot module c2900technology-package securityk9 R1(config)# end R1# copy running-config startup-configR1#reload
3. After the reloading is completed, issue the show version again to verify the securitytechnologypackagelicenseactivation.Doin RouterR3.

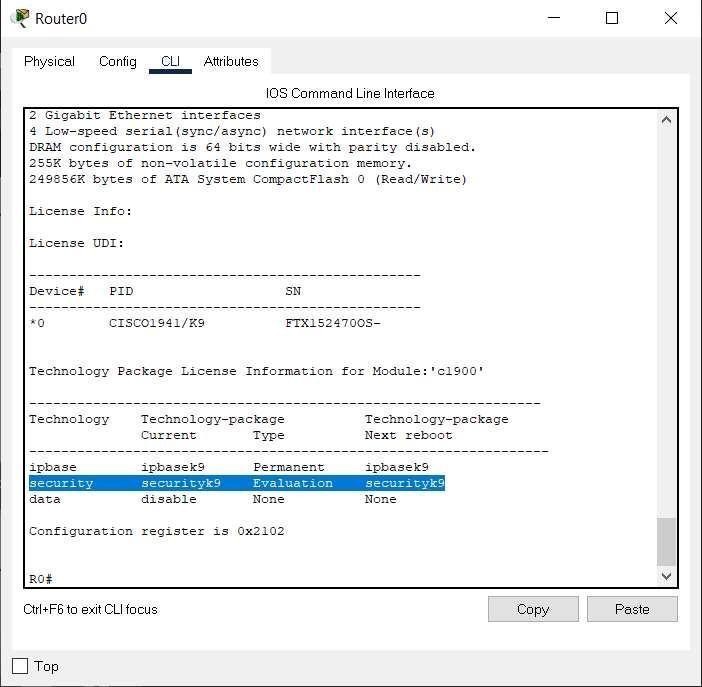
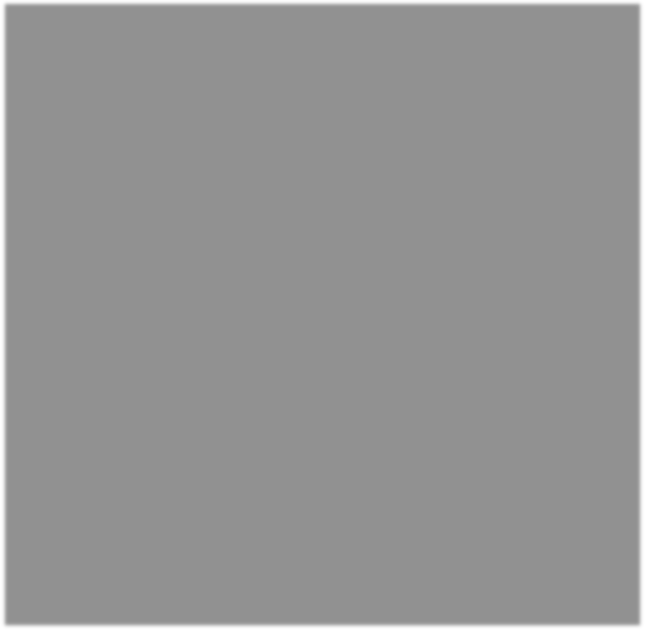
**SecurityisdisabledinRouter0**



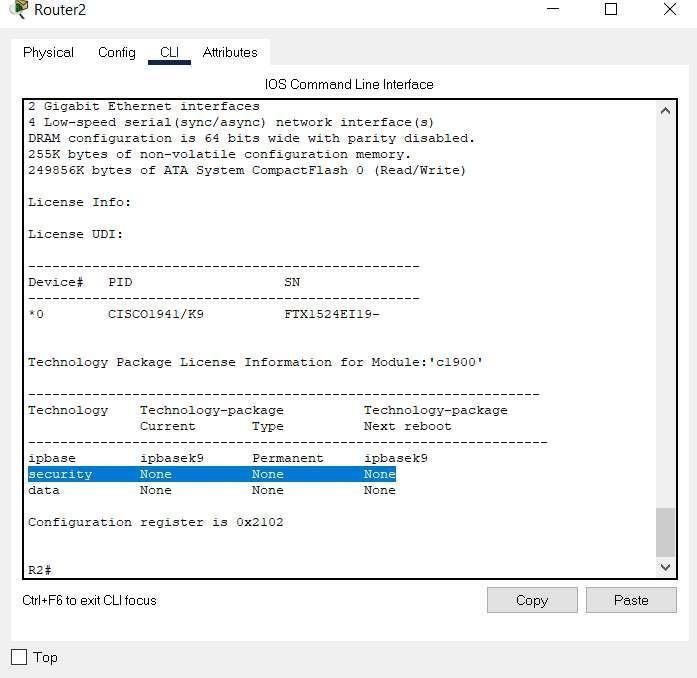
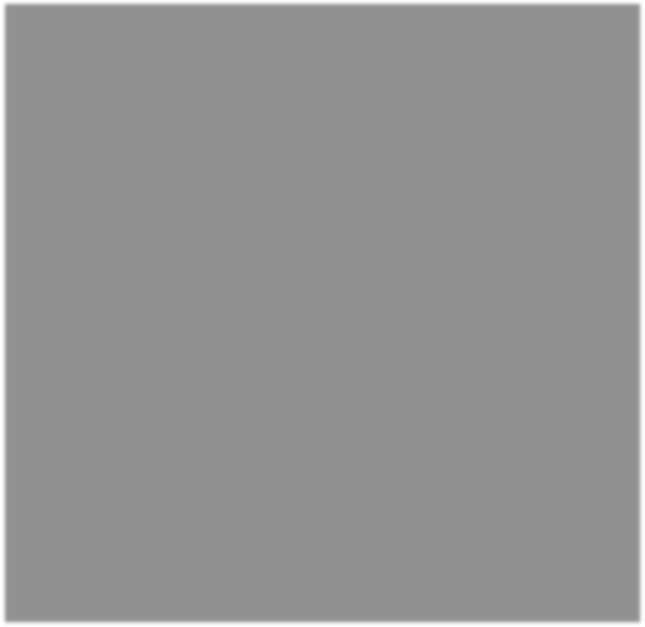
**EnablingSecurity**



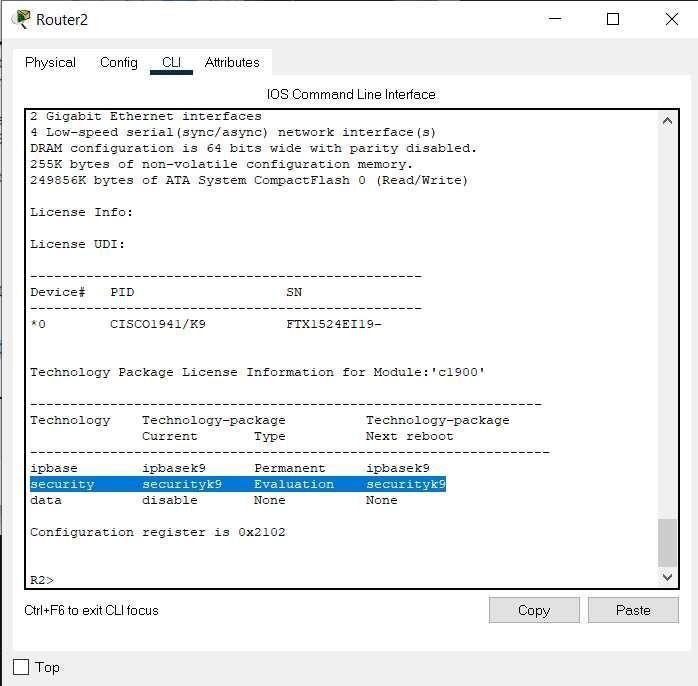
**Securityenabled**



**SecurityisdisabledinRouter0**



**Securityenabled**

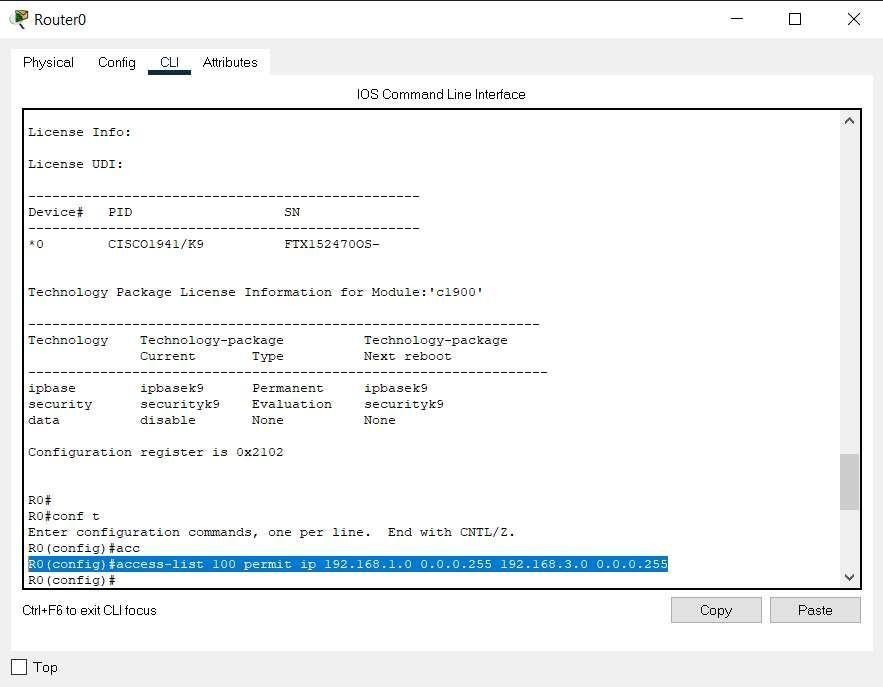
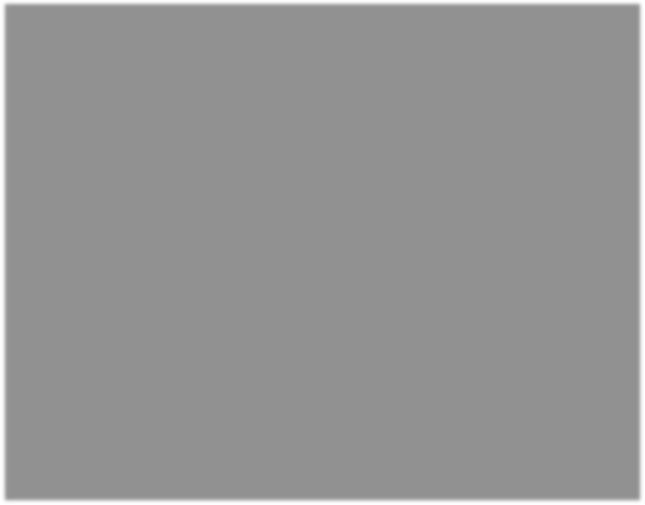


**Part2:ConfigureIPSecParametersonR1andR3**

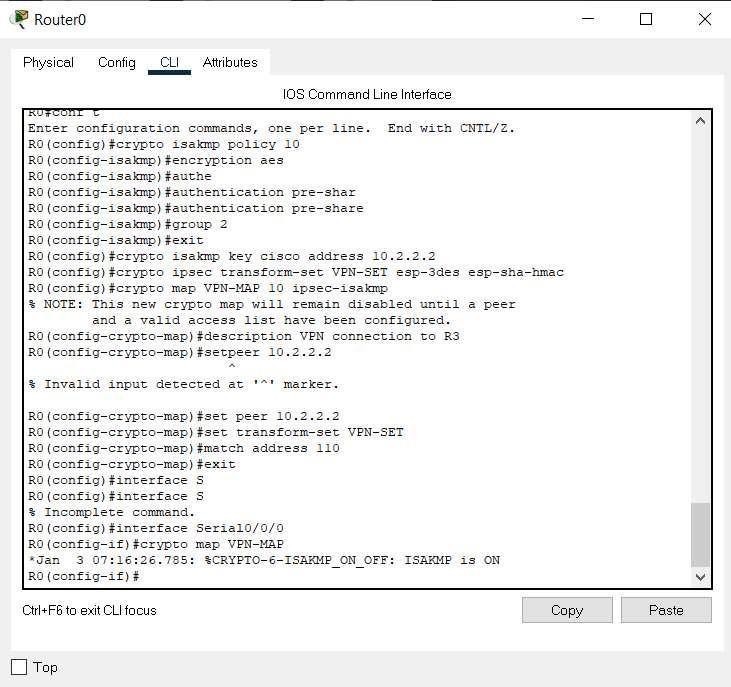
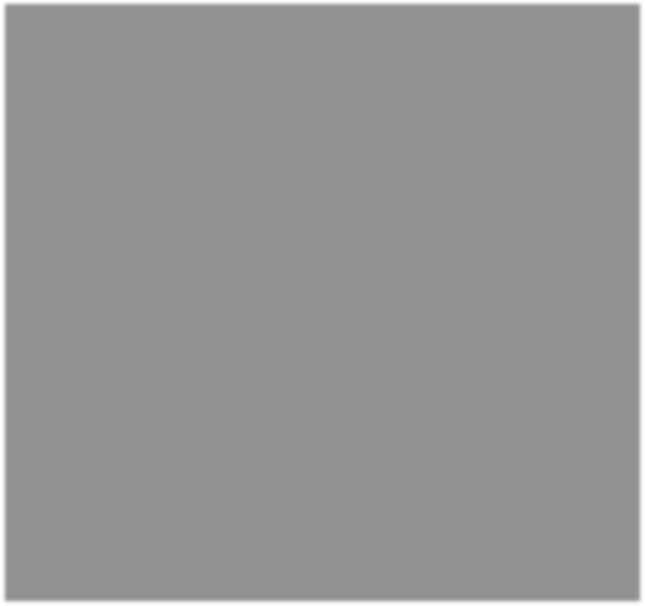
* 1. Testconnectivity
  2. IdentifyinterestingtrafficonR1.R1(config)#access-list110permitip192.168.1.00.0.0.255192.168.3.00.0.0.255

1. Configure the ISAKMP Phase 1 properties on R1. R1(config)# crypto isakmppolicy 10 R1(config-isakmp)# encryption aes R1(config-isakmp)#authentication preshare R1(config-isakmp)# group 2 R1(config-isakmp)# exitR1(config)# cryptoisakmp key ciscoaddress10.2.2.2
2. Configure the ISAKMP Phase 2 properties on R1. R1(config)# crypto ipsectransform-set VPN-SET esp-3des esp-sha-hmac R1(config)# crypto map VPN-MAP10ipsec-isakmpR1(config-crypto-map)#descriptionVPNconnectionto R3 R1(config-crypto-map)# set peer 10.2.2.2 R1(config-crypto-map)# settransform-set VPN-SET R1(config-crypto-map)# match address 110 R1(config-crypto-map)#exit
3. Configure the crypto map on the outgoing interface R1(config)# interfaceS0/0/0 R1(config-if)# cryptomap VPN-MAP
4. ConfigureIPSecParametersonR3 sameasR1

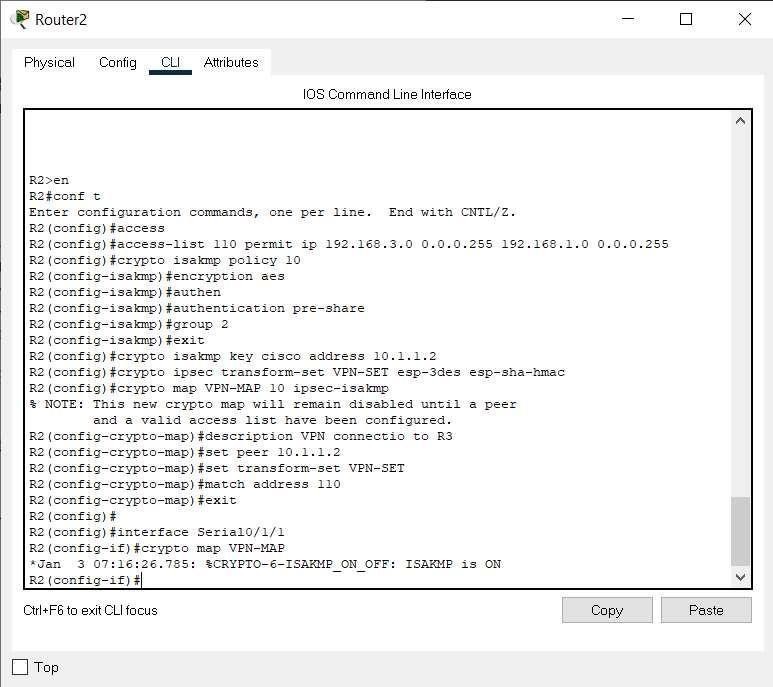
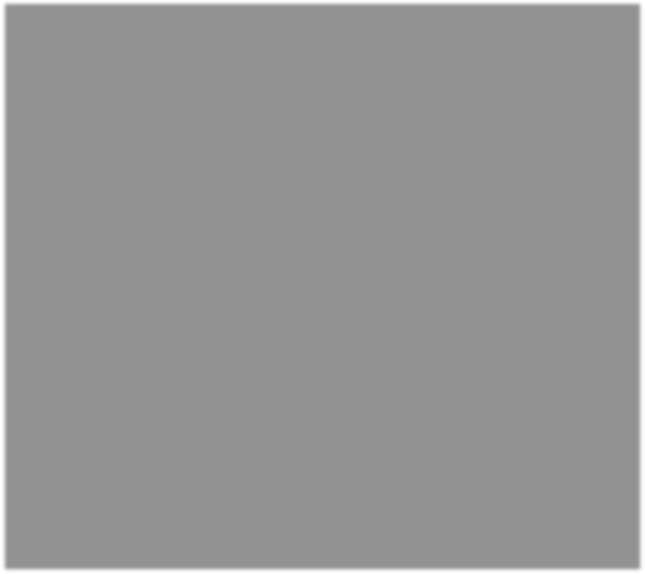
**Generatinginterestingtraffic**



**Executingpart2inRouter0**



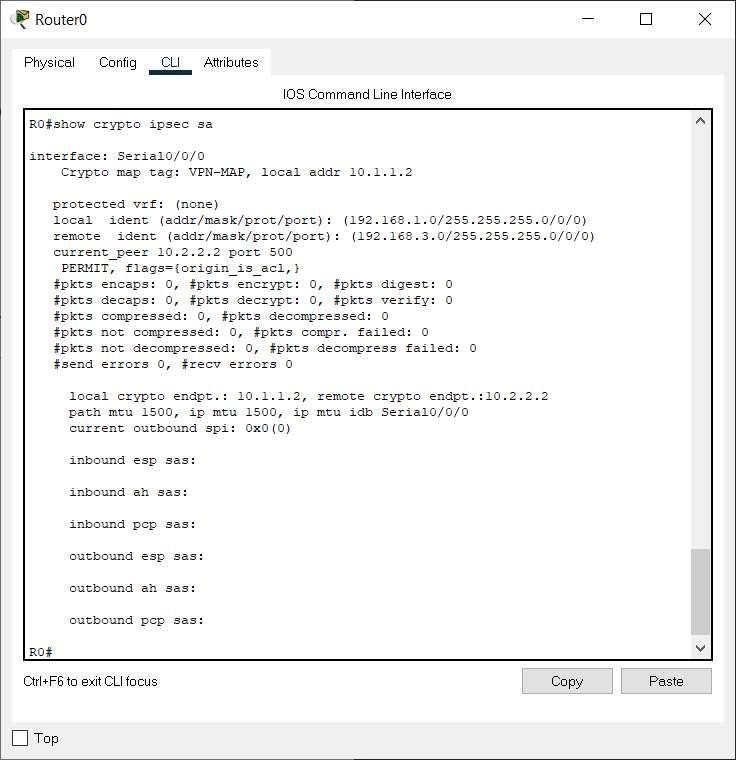
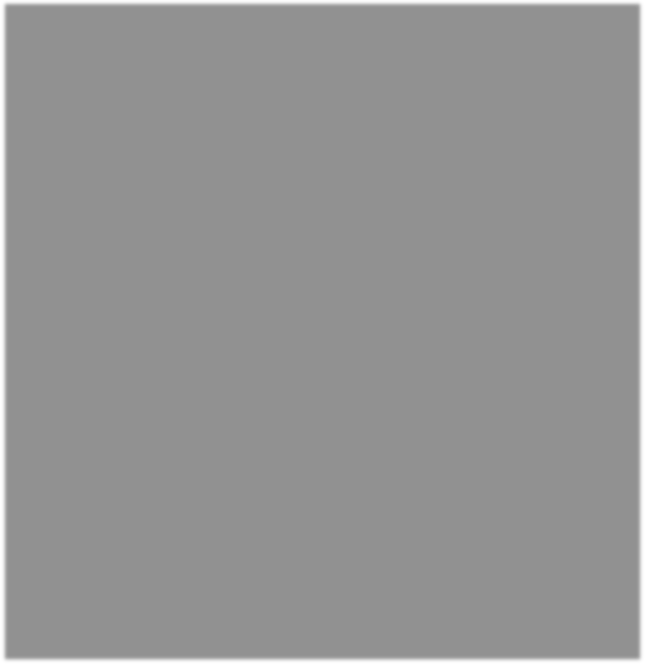
**Executingpart2inRouter2**



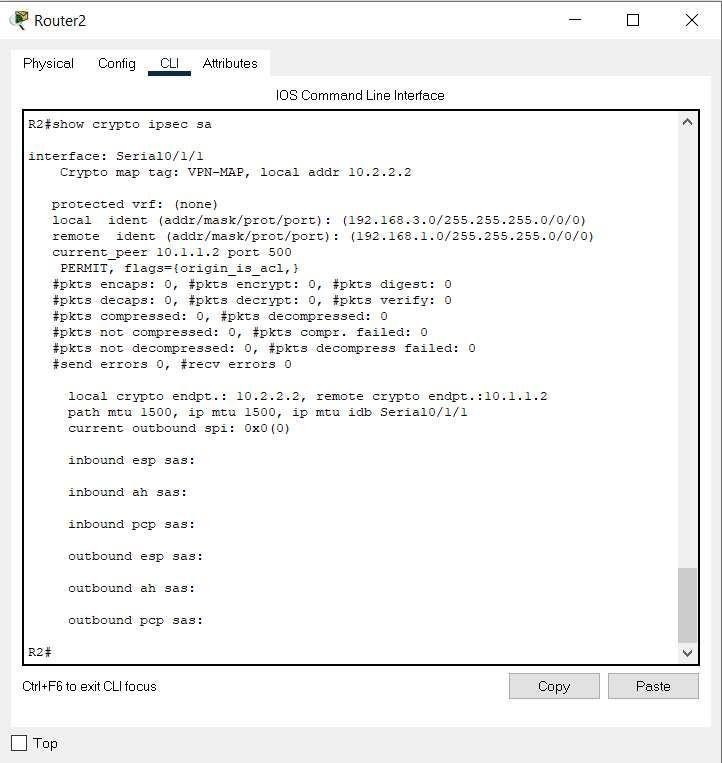
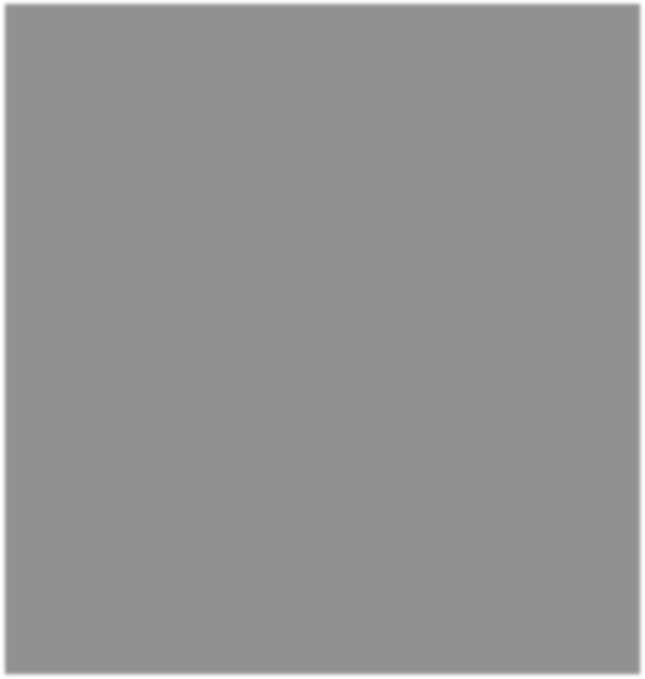
**Part3:VerifytheIPSecVPN**

1. Verifythetunnelpriortointerestingtraffic
2. R1#showcryptoipsecsa

**Executingpart3inRouter0**



**Executingpart3inRouter2**



### EXPERIMENT-10

**Aim:-**Create Self Signed Certificate and configure it for website.

**Tools / Apparatus:** O.S.: Microsoft Windows (any) / Linux / DOS

Packages: Turbo/Borland/GNU - C/C++

**Procedure:**

**To install a self-signed certificate in the Trusted Root Certification Authorities**

1. Open the certificate snap-in.

2. View certificates in the MMC snap-in

3. Select Run from the Start menu, and then enter mmc. The MMC appears.

4. From the File menu, select Add/Remove Snap In.

5. The Add or Remove Snap-ins window appears.

6. From the Available snap-ins list, choose Certificates, then select Add.

7. In the Certificates snap-in window, select Computer account, and then select Next.

8. Optionally, you can select My user account for the current user or Service account for a particular service.

9. In the Select Computer window, leave Local computer selected, and then select Finish.

10. In the Add or Remove Snap-in window, select OK.

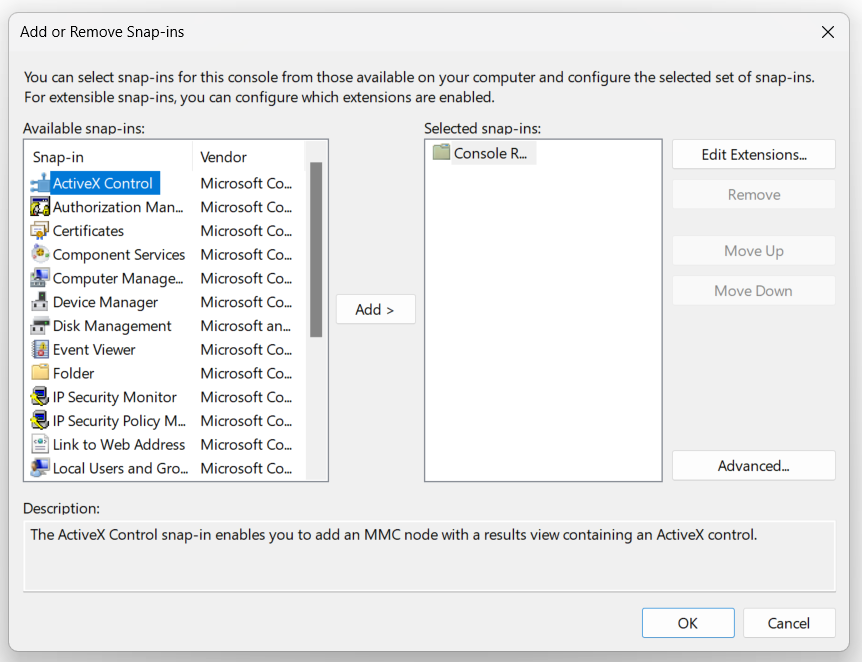
11. Open the folder to store the certificate, either the Local Computer or the Current User.

12. Open the Trusted Root Certification Authorities folder.

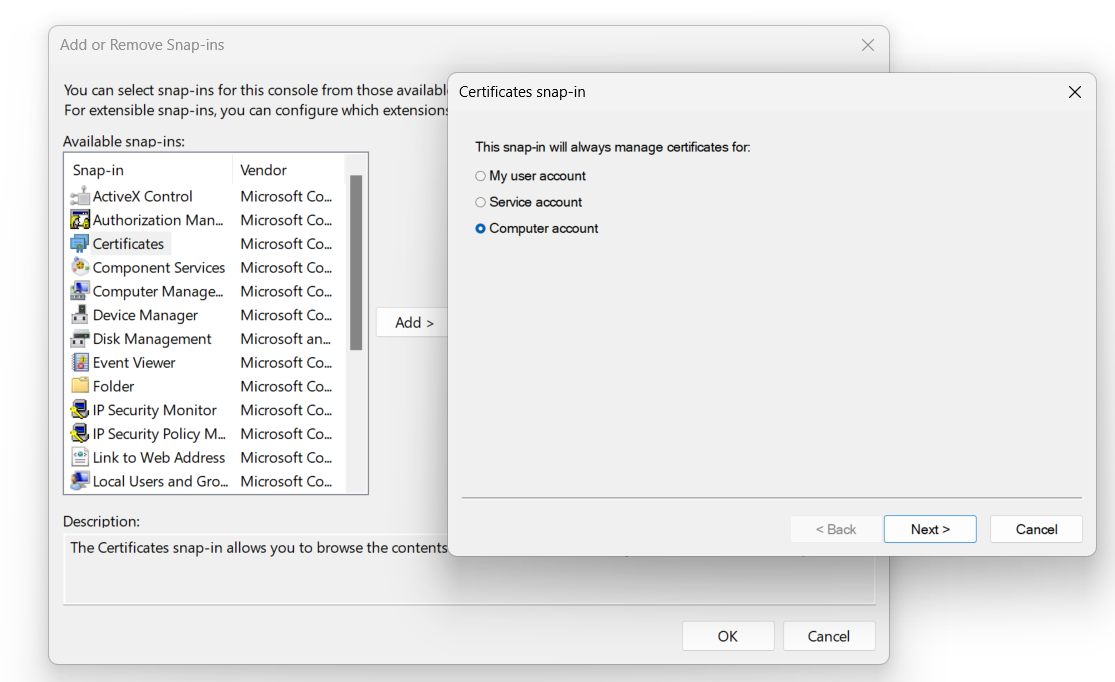
13. Right-click the Certificates folder and click All Tasks, then click Import.

14. Follow the on-screen wizard instructions to import the RootCA.pfx into the store.

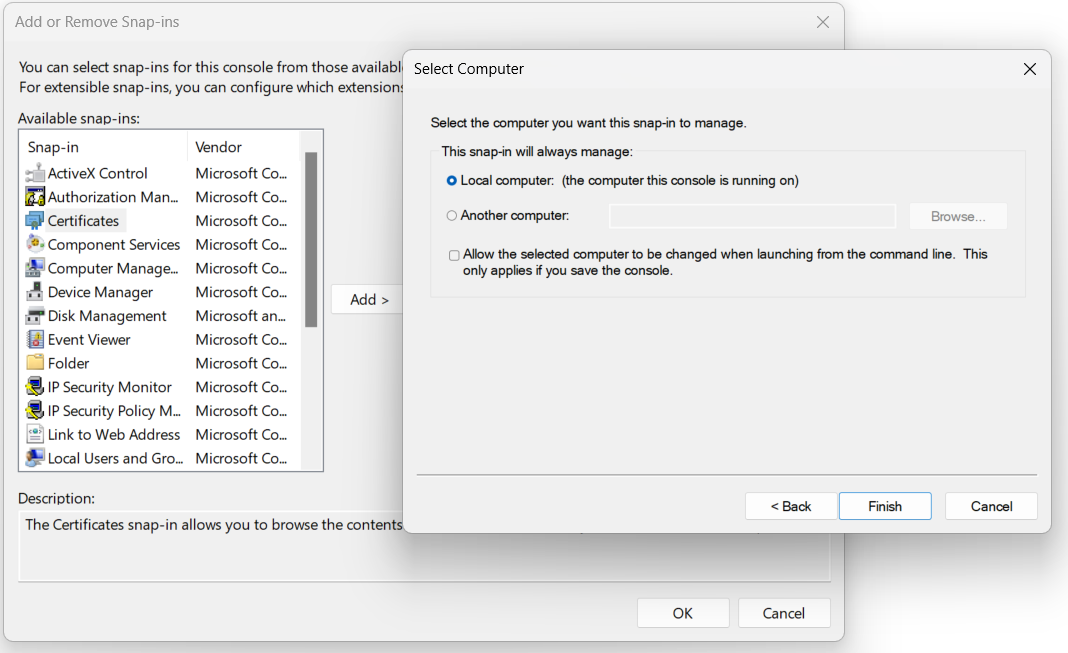
**Step 1:-**

****

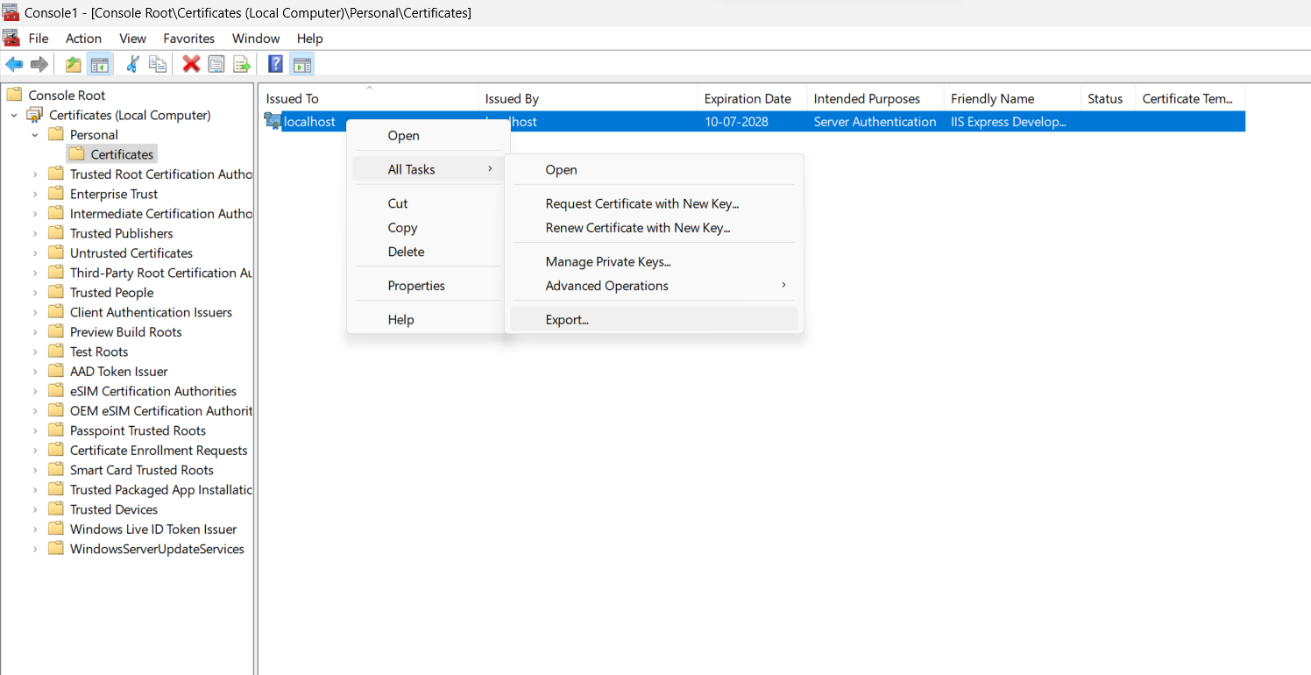
**Step 2:-**

****

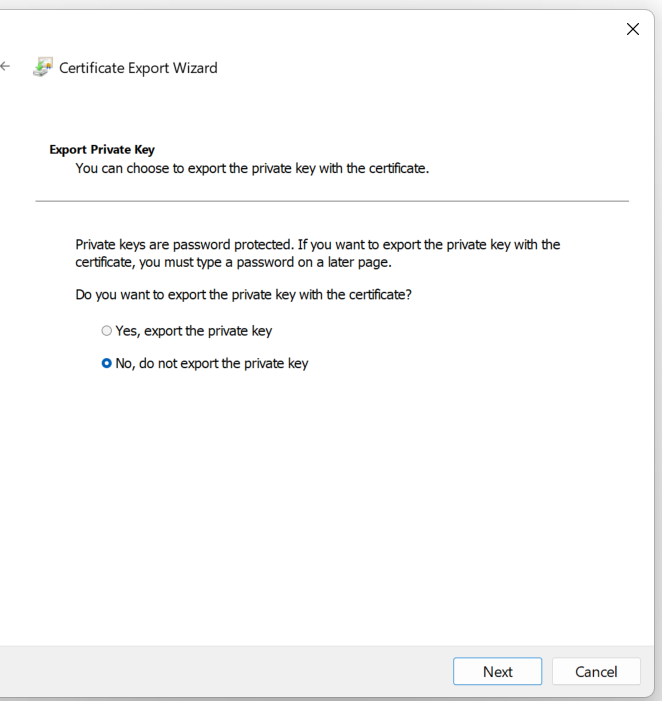
**Step 3:-**

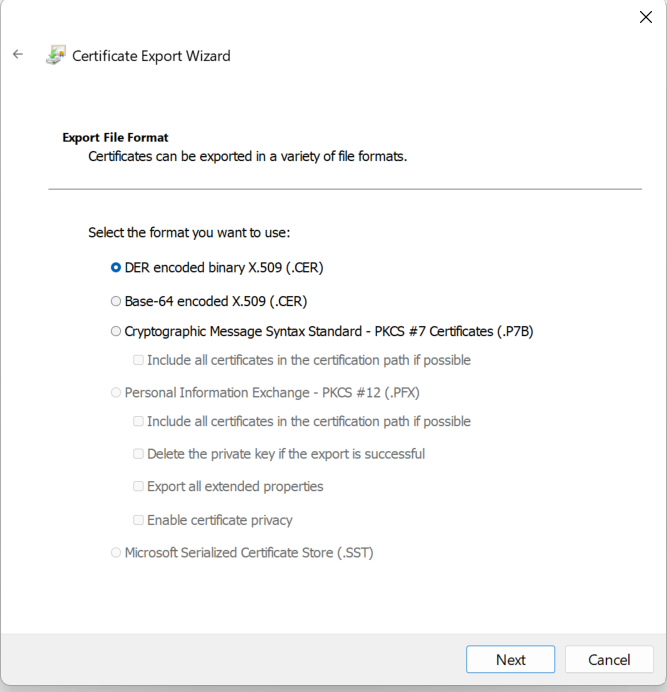
****

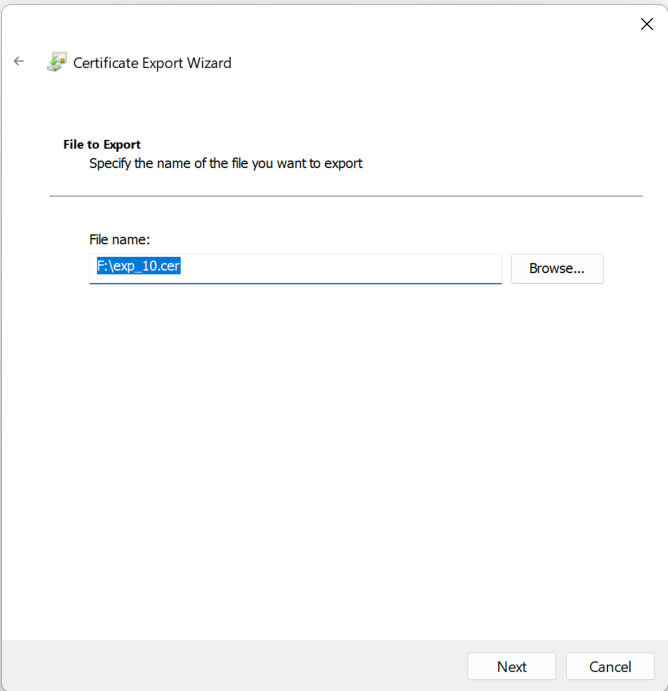
**Step 4:-**

****

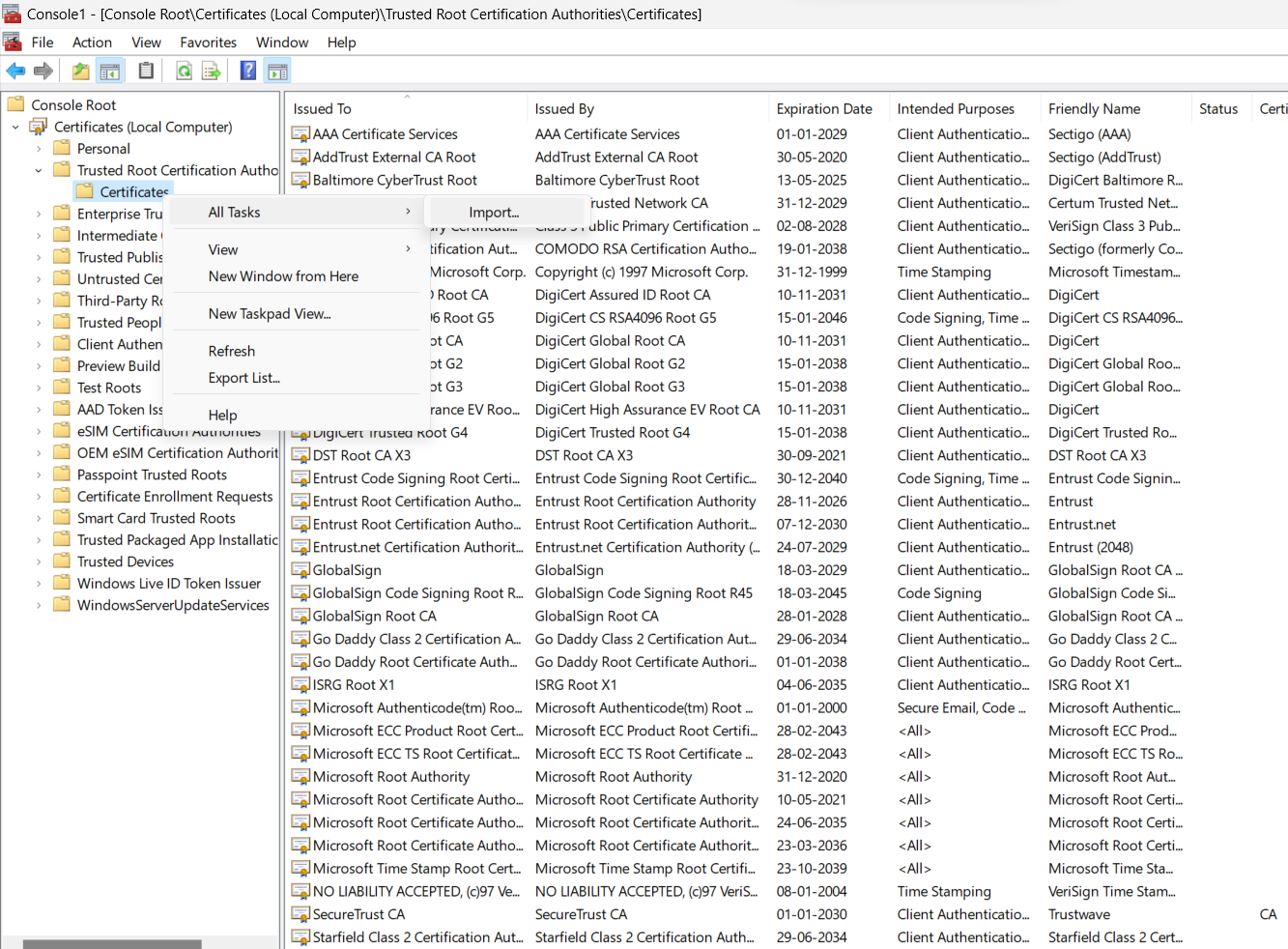
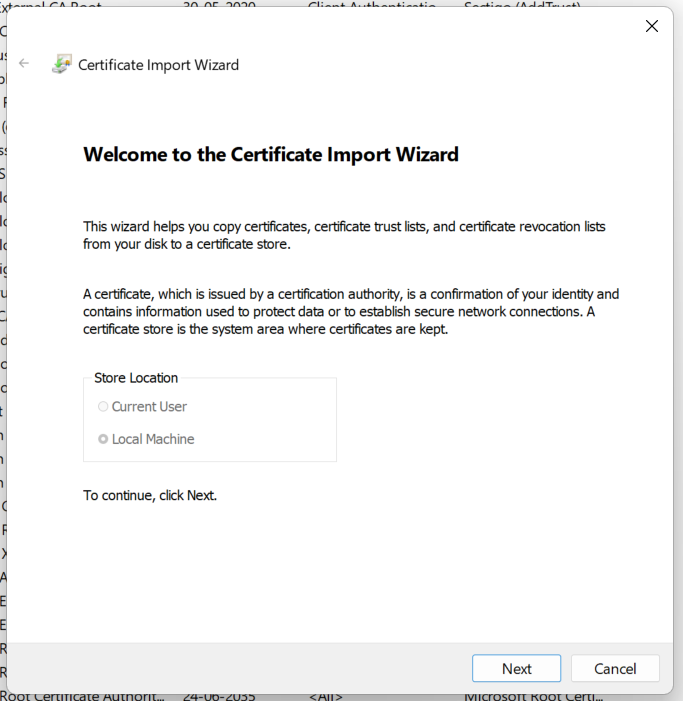
**Step 5:-**

****

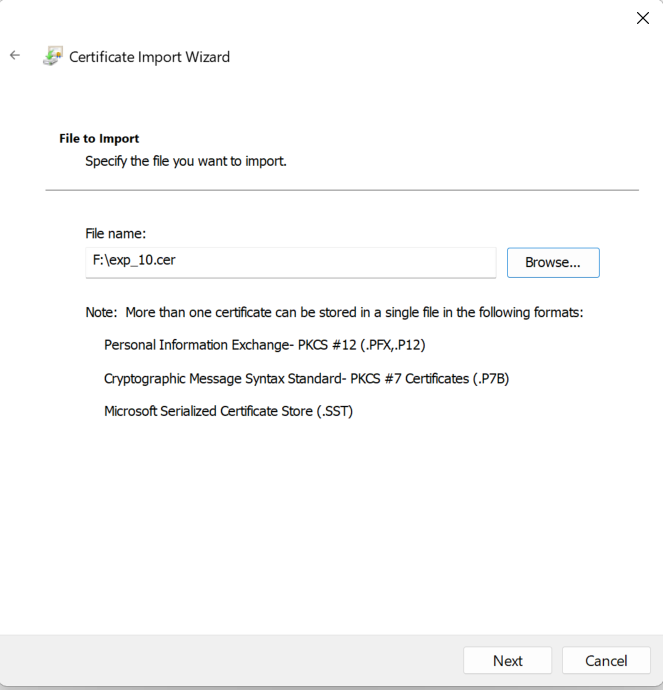
**Step 6:-**

**Step 7:- **

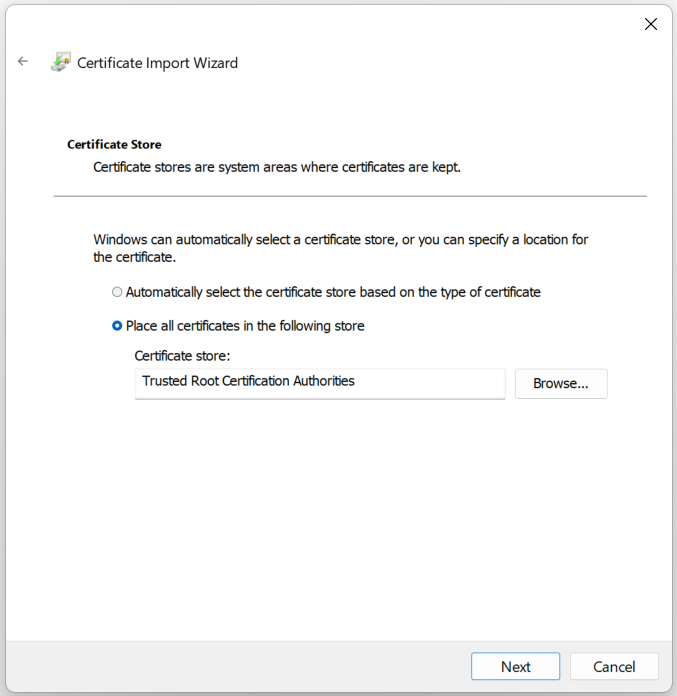
**Step 8:-**

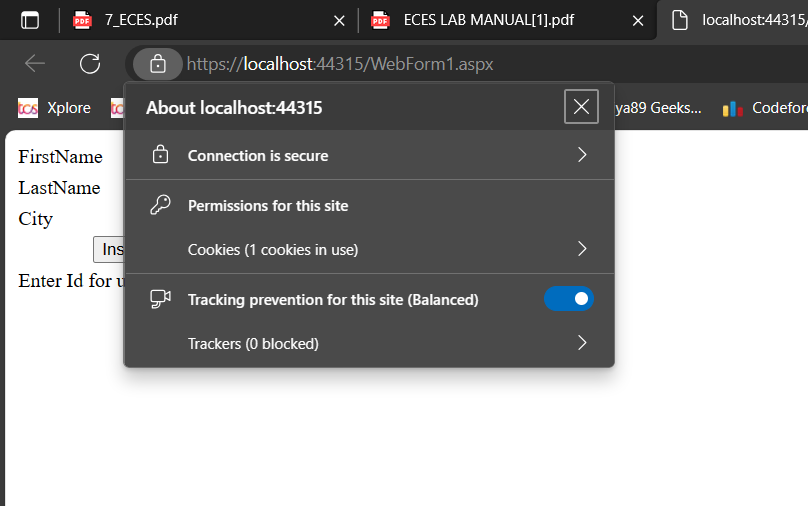
****

**Step 9:-**

**Step 10:-**

**Step 11:-**

****

**Output**:-