Ex No :

Date :

**Implement the following substitution and transportation techniques**

1. **Caesar cipher**

**Aim**

To write a program to implement Caesar cypher.

**Algorithm**

1. If the string is uppercase in order to encrypt it we use the formula.
2. Result = int(text[i]+s-65)%26+65
3. If it is a lowercase character the
4. Result = int(text[i]+s-97)%26+97
5. To decrypt the string if the character is uppercase the
6. Result = char(int(text[i]-s-97+26)%26+97)
7. If it is a lowercase the
8. Result = char(int(text[i]-k-65+26)%26+65)
9. Print the result

**Program**

#include<bits/stdc++.h>

using namespace std;

void comp(string str,int k)

{

int n=str.length();

string res="";

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

{

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

res+=char( (int(str[i])+k-97)%26 + 97 );

else

res+=char( (int(str[i])+k-65)%26 + 65 );

}

cout<<"Encrypted message : ";

cout<<res<<endl;

str="";

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

{

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

str+=char( (int(res[i])-k-97+26)%26 + 97 );

else

str+=char( (int(res[i])-k-65+26)%26 + 65 );

}

cout<<"Decrypted message : ";

cout<<str<<endl;

}

int main()

{

string str;

int k;

cout<<"Enter text and key value"<<endl;

cin>>str;

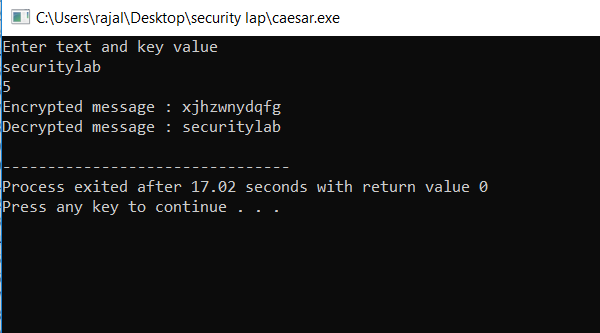
cin>>k;

comp(str,k);

return 0;

}

**Output**

****

1. **Playfair cipher**

**Aim**

To write a cpp program to implement the playfair cipher substitution technique.

**Algorithm**

1. Split the plain text into pairs of two letters. If an odd letter occurs, add ‘z’ to it.
2. If both letters are in same column take the letter below each one.
3. If both letters are in same row take the right one to each.
4. If neither of above are true form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

**Program**

#include <iostream>

#include <string>

using namespace std;

class playfair

{

public:

void doIt( string k, string t, bool ij, bool e )

{

createGrid( k, ij ); getTextReady( t, ij, e );

if( e ) doIt( 1 ); else doIt( -1 );

display();

}

private:

void doIt( int dir )

{

int a, b, c, d; string ntxt;

for( string::const\_iterator ti = \_txt.begin(); ti != \_txt.end(); ti++ )

{

if( getCharPos( \*ti++, a, b ) )

if( getCharPos( \*ti, c, d ) )

{

if( a == c ) { ntxt += getChar( a, b + dir ); ntxt += getChar( c, d + dir ); }

else if( b == d ){ ntxt += getChar( a + dir, b ); ntxt += getChar( c + dir, d ); }

else { ntxt += getChar( c, b ); ntxt += getChar( a, d ); }

}

}

\_txt = ntxt;

}

void display()

{

cout << "\n\n OUTPUT:\n=========" << endl;

string::iterator si = \_txt.begin(); int cnt = 0;

while( si != \_txt.end() )

{

cout << \*si; si++; cout << \*si << " "; si++;

if( ++cnt >= 26 ) cout << endl, cnt = 0;

}

cout << endl << endl;

}

char getChar( int a, int b )

{

return \_m[ (b + 5) % 5 ][ (a + 5) % 5 ];

}

bool getCharPos( char l, int &a, int &b )

{

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

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

if( \_m[y][x] == l )

{ a = x; b = y; return true; }

return false;

}

void getTextReady( string t, bool ij, bool e )

{

for( string::iterator si = t.begin(); si != t.end(); si++ )

{

\*si = toupper( \*si ); if( \*si < 65 || \*si > 90 ) continue;

if( \*si == 'J' && ij ) \*si = 'I';

else if( \*si == 'Q' && !ij ) continue;

\_txt += \*si;

}

if( e )

{

string ntxt = ""; size\_t len = \_txt.length();

for( size\_t x = 0; x < len; x += 2 )

{

ntxt += \_txt[x];

if( x + 1 < len )

{

if( \_txt[x] == \_txt[x + 1] ) ntxt += 'X';

ntxt += \_txt[x + 1];

}

}//OR KD BX GA TL KS ML HP LK HY AN PM QG

\_txt = ntxt;

}

if( \_txt.length() & 1 ) \_txt += 'X';

}

void createGrid( string k, bool ij )

{

if( k.length() < 1 ) k = "KEYWORD";

k += "ABCDEFGHIJKLMNOPQRSTUVWXYZ"; string nk = "";

for( string::iterator si = k.begin(); si != k.end(); si++ )

{

\*si = toupper( \*si ); if( \*si < 65 || \*si > 90 ) continue;

if( ( \*si == 'J' && ij ) || ( \*si == 'Q' && !ij ) )continue;

if( nk.find( \*si ) == -1 ) nk += \*si;

}

copy( nk.begin(), nk.end(), &\_m[0][0] );

}

string \_txt; char \_m[5][5];

};

int main( int argc, char\* argv[] )

{

string key, i, txt; bool ij, e;

cout << "(E)ncode or (D)ecode? "; getline( cin, i ); e = ( i[0] == 'e' || i[0] == 'E' );

cout << "Enter a en/decryption key: "; getline( cin, key );

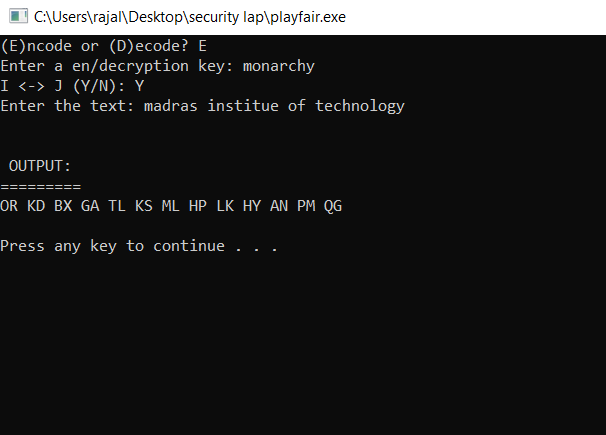
cout << "I <-> J (Y/N): "; getline( cin, i ); ij = ( i[0] == 'y' || i[0] == 'Y' );

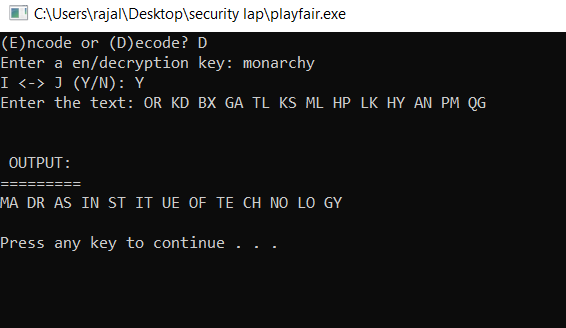
cout << "Enter the text: "; getline( cin, txt );

playfair pf; pf.doIt( key, txt, ij, e ); return system( "pause" );

}

**Output**

****

****

1. **Hill cipher**

**Aim**

To write a cpp program to implement the hill cipher substitution technique.

**Algorithm**

The given message string and key string should be represented as matrix.

The key and message string are multiplied

Finally modulo 26 is taken for each element of the matrix.

The letters for the corresponding numbers are taken and these are encrypted text is generated.

Step 3 and step 4 are done to get the decrypted text.

**Program**

#include<iostream>

#include<math.h>

using namespace std;

float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];

void encryption(); //encrypts the message

void decryption(); //decrypts the message

void getKeyMessage(); //gets key and message from user

void inverse(); //finds inverse of key matrix

int main() {

getKeyMessage();

encryption();

decryption();

}

void encryption() {

int i, j, k;

for(i = 0; i < 3; i++)

for(j = 0; j < 1; j++)

for(k = 0; k < 3; k++)

encrypt[i][j] = encrypt[i][j] + a[i][k] \* mes[k][j];

cout<<"\nEncrypted string is: ";

for(i = 0; i < 3; i++)

cout<<(char)(fmod(encrypt[i][0], 26) + 97);

}

void decryption() {

int i, j, k;

inverse();

for(i = 0; i < 3; i++)

for(j = 0; j < 1; j++)

for(k = 0; k < 3; k++)

decrypt[i][j] = decrypt[i][j] + b[i][k] \* encrypt[k][j];

cout<<"\nDecrypted string is: ";

for(i = 0; i < 3; i++)

cout<<(char)(fmod(decrypt[i][0], 26) + 97);

cout<<"\n";

}

void getKeyMessage() {

int i, j;

char msg[3];

cout<<"Enter 3x3 matrix for key (It should be inversible):\n";

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++) {

scanf("%f", &a[i][j]);

c[i][j] = a[i][j];

}

cout<<"\nEnter a 3 letter string: ";

cin>>msg;

for(i = 0; i < 3; i++)

mes[i][0] = msg[i] - 97;

}

void inverse() {

int i, j, k;

float p, q;

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++) {

if(i == j)

b[i][j]=1;

else

b[i][j]=0;

}

for(k = 0; k < 3; k++) {

for(i = 0; i < 3; i++) {

p = c[i][k];

q = c[k][k];

for(j = 0; j < 3; j++) {

if(i != k) {

c[i][j] = c[i][j]\*q - p\*c[k][j];

b[i][j] = b[i][j]\*q - p\*b[k][j];

}

}

}

}

for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++)

b[i][j] = b[i][j] / c[i][i];

cout<<"\n\nInverse Matrix is:\n";

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++)

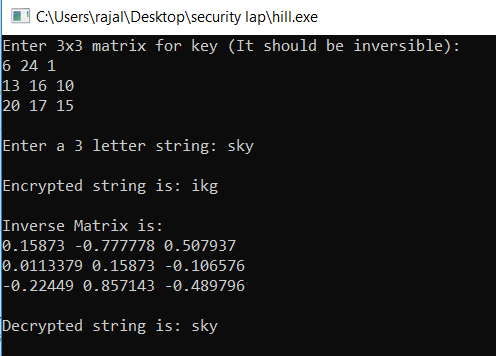
cout<<b[i][j]<<" ";

cout<<"\n";

}

}

**Output**

****

1. **Vignere cipher**

**Aim**

To write a cpp program to implement vignere cipher.

**Algorithm**

1. The encryption of original text is done by using vignere square of the table.
2. The table consists of the alphabets written out 26 times in different rows each alphabet shifted cyclically to the left compared to the previous alphabet
3. At different points in the encryption process the cipher user a different alphabet from one of the rows.
4. The alphabet used at each point depends on the repeating key words

**Program**

#include <iostream>

#include <string>

using namespace std;

class Vigenere

{

public:

string key;

Vigenere(string key)

{

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

{

if (key[i] >= 'A' && key[i] <= 'Z')

this->key += key[i];

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

this->key += key[i] + 'A' - 'a';

}

}

string encrypt(string text)

{

string out;

for (int i = 0, j = 0; i < text.length(); ++i)

{

char c = text[i];

if (c >= 'a' && c <= 'z')

c += 'A' - 'a';

else if (c < 'A' || c > 'Z')

continue;

out += (c + key[j] - 2 \* 'A') % 26 + 'A';

j = (j + 1) % key.length();

}

return out;

}

string decrypt(string text)

{

string out;

for (int i = 0, j = 0; i < text.length(); ++i)

{

char c = text[i];

if (c >= 'a' && c <= 'z')

c += 'A' - 'a';

else if (c < 'A' || c > 'Z')

continue;

out += (c - key[j] + 26) % 26 + 'A';

j = (j + 1) % key.length();

}

return out;

}

};

int main()

{

cout<<"Enter key"<<endl;

string key;

cin>>key;

Vigenere cipher(key);

string text;

cout<<"Enter the text"<<endl;

cin>>text;

string encrypted = cipher.encrypt(text);

string decrypted = cipher.decrypt(encrypted);

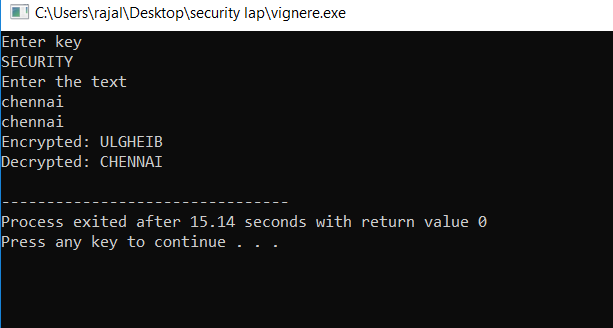
cout << text << endl;

cout << "Encrypted: " << encrypted << endl;

cout << "Decrypted: " << decrypted << endl;

}

**Output**

****

1. **Rail fence cipher**

**Aim**

To write a cpp program to implement railfence cipher

**Algorithm**

1. In a transposition cipher the order of the alphabets is rearranged to obtain the cipher text.
2. In the rail fence cipher the plain text is written downloads and diagonally on successive rails of an imaginary fence.
3. After each alphabet has been written the individual rows are combined to obtain the cipher text.

**Program**

#include <bits/stdc++.h>

using namespace std;

string encryptRailFence(string text, int key)

{

char rail[key][(text.length())];

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

for (int j = 0; j < text.length(); j++)

rail[i][j] = '\n';

bool dir\_down = false;

int row = 0, col = 0;

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

{

if (row == 0 || row == key-1)

dir\_down = !dir\_down;

rail[row][col++] = text[i];

dir\_down?row++ : row--;

}

string result;

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

for (int j=0; j < text.length(); j++)

if (rail[i][j]!='\n')

result.push\_back(rail[i][j]);

return result;

}

string decryptRailFence(string cipher, int key)

{

char rail[key][cipher.length()];

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

for (int j=0; j < cipher.length(); j++)

rail[i][j] = '\n';

bool dir\_down;

int row = 0, col = 0;

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

{

if (row == 0)

dir\_down = true;

if (row == key-1)

dir\_down = false;

rail[row][col++] = '\*';

dir\_down?row++ : row--;

}

int index = 0;

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

for (int j=0; j<cipher.length(); j++)

if (rail[i][j] == '\*' && index<cipher.length())

rail[i][j] = cipher[index++];

string result;

row = 0, col = 0;

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

{

if (row == 0)

dir\_down = true;

if (row == key-1)

dir\_down = false;

if (rail[row][col] != '\*')

result.push\_back(rail[row][col++]);

dir\_down?row++: row--;

}

return result;

}

int main()

{

cout<<"Enter the text and key"<<endl;

string str;

int key;

cin>>str>>key;

string enc = encryptRailFence(str,key) ;

cout<<endl;

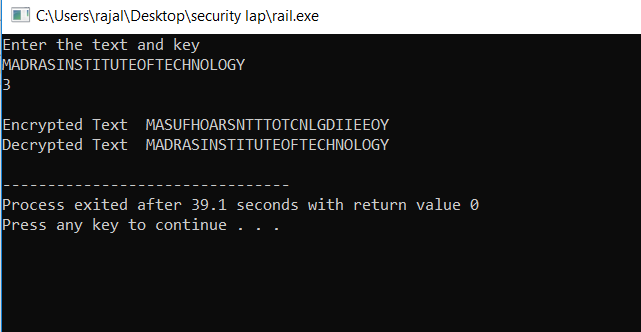
cout<<"Encrypted Text "<<enc<<endl;

cout<<"Decrypted Text "<<decryptRailFence(enc,key)<<endl;

return 0;

}

**Output**



**Result**

All the above ciphers (Caesar cipher, paly fair cipher, hill cipher, vignere cipher, railfence cipher ) are compiled successfully and output verified.

Ex No :

Date :

**Implementing the following algorithms**

1. **DES**
2. **RSA**
3. **Diffie-Hellman**
4. **MD5**
5. **SHA 1**
6. **DES**

**Aim**

To understand and write a program for DES.

**Algorithm**

1. DES is a block cipher symmetric key
2. First of all we need to get the key generation instance using DES algorithm
3. Generate secretly used to encrypt and decrypt.
4. Fragmenting of the text into 8 octal 64 bits blocks
5. Initial permutation of the blocks , blocks of blocks into two parts left and right named L and R
6. Permutation and substitution repeated 16 times
7. Rejoin the left and right parts then inverse initial permutation
8. Create 16 subkeys each of which is 48 bits long
9. Total 16 rounds

**Program**

#include <bits/stdc++.h>

using namespace std;

string Permutation(vector<int> array, string inp){

string out = "";

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

out += inp[array[i]-1];

return out;

}

class S\_DES{

public:

string KEY,K1,K2,IPOut,InvIPOut;

string F1Out;

string INPUT,OUTPUT;

void initialize(string key){

if(key.size()!=10){

cout<<"\nInValid Key-Length "<<key<<" "<<key.size();

exit(1);

}

KEY = key;

Keys\_Generation();

}

void Keys\_Generation(){

vector<int> P10(10);

int sak[10]={3 ,5, 2, 7, 4, 10, 1, 9, 8, 6};

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

P10[i]=sak[i];

string P10\_output = Permutation(P10,KEY);

cout<<"P10 output while generating key: "<<P10\_output<<endl;

string P10\_left = P10\_output.substr(0,5), P10\_right = P10\_output.substr(5,5);

string pl = LShift(P10\_left,1), pr = LShift(P10\_right,1);

string plpr = pl+pr;

vector<int> P8(10,0);

int sak1[8]={6, 3, 7, 4, 8, 5, 10, 9};

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

P8[i]=sak1[i];

K1 = Permutation(P8,plpr);

cout<<"K1: "<<K1<<endl;

string pl1=LShift(pl,2), pr1=LShift(pr,2);

plpr = pl1+pr1;

K2 = Permutation(P8,plpr);

cout<<"K2: "<<K2<<endl;

}

string LShift(string input,int n){

string output = input;

char firstbit;

while(n--){

firstbit = output[0];

output = output.substr(1,output.size()-1);

output += firstbit;

}

return output;

}

void DES\_Encryption(){

IP();

string LIP = IPOut.substr(0,4);

string RIP = IPOut.substr(4,4);

cout<<"IP output: "<<IPOut<<endl;

Function\_F(LIP,RIP,1);

cout<<"Fn Output: "<<F1Out<<endl;

string L1 = F1Out.substr(0,4), R1 = F1Out.substr(4,4);

Function\_F(R1,L1,2);

cout<<"Fn Output second time: "<<F1Out<<endl;

InvIP(F1Out);

cout<<"Encrypted Cipher-string: "<<InvIPOut<<endl;

}

/\*Method to perform Initial-Permutation\*/

void IP(){

vector<int> IP\_array(8,0);

int number[8]={2, 6, 3, 1, 4, 8, 5, 7};

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

IP\_array[i]=number[i];

IPOut = Permutation(IP\_array,INPUT);

}

/\*Method to perform Inverse of Initial-Permutation\*/

void InvIP(string input){

vector<int> InvIPArray(8,0);

int number[8]={4, 1, 3, 5, 7, 2, 8, 6};

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

InvIPArray[i]=number[i];

InvIPOut = Permutation(InvIPArray,input);

}

void Function\_F(string linput,string rinput,int key)

{

vector<int> E\_P(8,0);

int nn2[8]={4, 1, 2, 3, 2, 3, 4, 1};

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

E\_P[i]==nn2[i];

string E\_POutput = Permutation(E\_P,rinput);

string EXOR\_Output;

if(key == 1)

EXOR\_Output = EX\_OR(E\_POutput,K1);

else

EXOR\_Output = EX\_OR(E\_POutput,K2);

string LEXOR = EXOR\_Output.substr(0,4),REXOR = EXOR\_Output.substr(4,4);

string SBOX0\_Output=SBOX0(LEXOR);

string SBOX1\_Output=SBOX1(REXOR);

string SBOX\_Output = SBOX0\_Output+SBOX1\_Output;

vector<int> P4(4,0);

int v4[4]={2, 4, 3, 1};

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

P4[i]=v4[i];

string P4\_Output = Permutation(P4,SBOX\_Output);

string fk\_Output = EX\_OR(P4\_Output,linput);

F1Out = fk\_Output + rinput;

}

string EX\_OR(string a,string b){

string output = "";

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

if(a[i] == b[i])

output += "0";

else

output += "1";

}

return output;

}

string SBOX0(string l)

{

vector<int> temp(4,0);

vector<vector<int> > S0(4,temp);

int number[4][4]={

{1, 0, 3, 2},

{3, 2, 1, 0},

{0, 2 ,1, 3},

{3, 1, 3, 2}

};

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

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

S0[i][j]=number[i][j];

}

string bits[]={"00","01","10","11"};

string lrow = l.substr(0,1)+l.substr(3,1),lcol = l.substr(1,1)+l.substr(2,1);

string SO;

int i,lr,lc,b;

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

if(lrow == bits[i])

lr=i;

if(lcol == bits[i])

lc=i;

}

b=S0[lr][lc];

return bits[b];

}

string SBOX1(string l)

{

vector<int> temp(4,0);

vector<vector<int> > S0(4,temp);

int number[4][4]={

{0, 1, 2, 3},

{2, 0, 1, 3},

{3, 0 ,1, 0},

{2, 1, 0, 3}

};

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

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

S0[i][j]=number[i][j];

}

string bits[]={"00","01","10","11"};

string lrow = l.substr(0,1)+l.substr(3,1),lcol = l.substr(1,1)+l.substr(2,1);

string SO;

int i,lr,lc,b;

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

if(lrow == bits[i])

lr=i;

if(lcol == bits[i])

lc=i;

}

b=S0[lr][lc];

return bits[b];

}

};

int main()

{

int i,n=10,choice;

string key;

S\_DES S;

cout<<"\nEnter the 10-bits KEY: ";

cin>>key;

cout<<"\nNotedown this key, as same key is used for Decryption\n";

S.initialize(key);

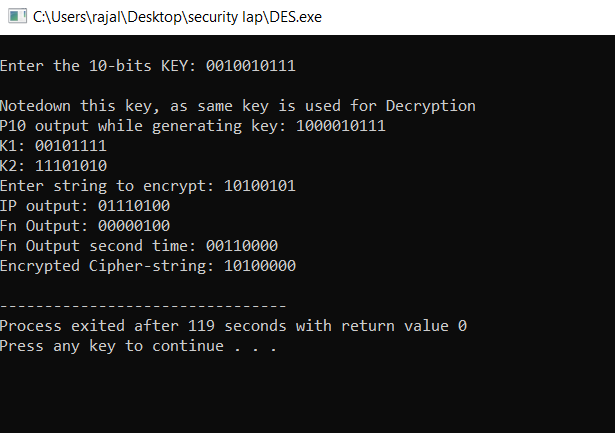
cout<<"Enter string to encrypt: ";

cin>>S.INPUT;

S.DES\_Encryption();

return 0;

}



1. RSA

**Aim**

To write a program for implementing RSA algorithm

**Algorithm**

**Program**

#include <bits/stdc++.h>

using namespace std;

int gcd(int a, int h)

{

int temp;

while (1)

{

temp = a%h;

if (temp == 0)

return h;

a = h;

h = temp;

}

}

int main()

{

double p;

double q;

cout<<"Enter first prime number"<<endl;

cin>>p;

cout<<"Enter second prime number"<<endl;

cin>>q;

double n = p\*q;

double e = 2;

double phi = (p-1)\*(q-1);

while (e < phi)

{

if (gcd(e, phi)==1)

break;

else

e++;

}

int k = 2;

double d = (1 + (k\*phi))/e;

double msg;

cout<<"Enter message data"<<endl;

cin>>msg;

printf("Message data = %lf", msg);

double c = pow(msg, e);

c = fmod(c, n);

printf("\nEncrypted data = %lf", c);

double m = pow(c, d);

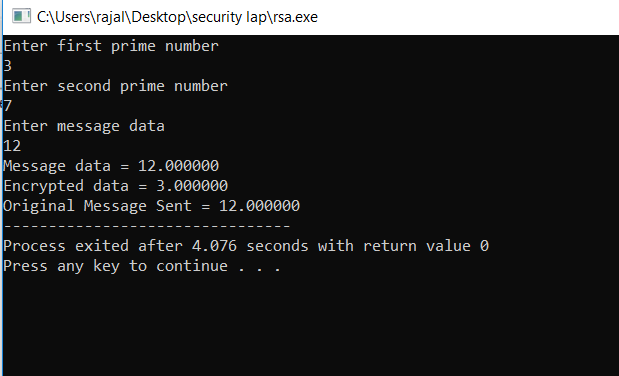
m = fmod(m, n);

printf("\nOriginal Message Sent = %lf", m);

return 0;

}

**Output**

****

**C.Diffie hellman**

**Aim**

To write a program to implement Diffie hellman.

**Algorithm**

* For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables one prime P and G (a primitive root of P) and two private values a and b.
* P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly, the opposite person received the key and from that generates a secret key after which they have the same secret key to encrypt.

**Program**

#include<bits/stdc++.h>

using namespace std;

// Power function to return value of a ^ b mod P

long long int power(long long int a, long long int b,

long long int P)

{

if (b == 1)

return a;

else

return (((long long int)pow(a, b)) % P);

}

int main()

{

long long int P, G, x, a, y, b, ka, kb;

cout<<"Enter number P"<<endl;

cin>>P;

printf("The value of P : %lld\n", P);

cout<<"Enter number G"<<endl;

cin>>G;

printf("The value of G : %lld\n\n", G);

cout<<"Enter number a"<<endl;

cin>>a;

printf("The private key a for Alice : %lld\n", a);

x = power(G, a, P);

cout<<"Enter number b"<<endl;

cin>>b;

printf("The private key b for Bob : %lld\n\n", b);

y = power(G, b, P);

ka = power(y, a, P);

kb = power(x, b, P);

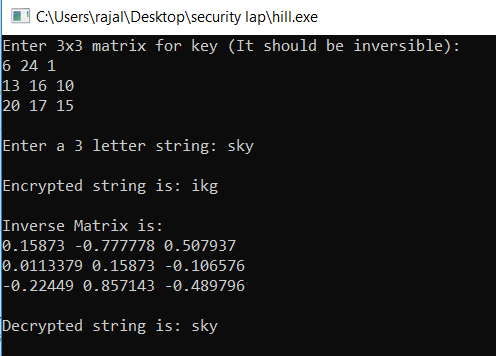
printf("Secret key for the Alice is : %lld\n", ka);

printf("Secret Key for the Bob is : %lld\n", kb);

return 0;

}

**Output**



1. **MD5**

**Aim**

To write a program to implement MD5.

**Algorithm**

1. Divides the input in blocks of 32 bits each.
2. 64 bits are inserted at end of last block
3. There are 64 bits record the length of original input
4. MD5 help function

**Program**

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class MD5 {

public static String getMd5(String input)

{

try {

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

byte[] messageDigest = md.digest(input.getBytes());

BigInteger no = new BigInteger(1, messageDigest);

String hashtext = no.toString(16);

while (hashtext.length() < 32) {

hashtext = "0" + hashtext;

}

return hashtext;

}

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

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

{

String s = "GeeksForGeeks";

System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));

}

}

**Output**

Enter the text :

hello world

MD5 value :

5eb63bbbe01eeed093cb22bb8f5acdc3

1. **SHA 1**

**Aim**

To write a program to implement SHA1.

**Algorithm**

The Secure Hash Algorithm 1 (SHA-1) is a cryptographic computer security algorithm. It was created by the US National Security Agency in 1995, after the SHA-0 algorithm in 1993, and it is part of the Digital Signature Algorithm or the Digital Signature Standard (DSS).

**Program**

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class SHA1 {

private static String convertToHex(byte[] data) {

StringBuffer buf = new StringBuffer();

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

int halfbyte = (data[i] >>> 4) & 0x0F;

int two\_halfs = 0;

do {

if ((0 <= halfbyte) && (halfbyte <= 9))

buf.append((char) ('0' + halfbyte));

else

buf.append((char) ('a' + (halfbyte - 10)));

halfbyte = data[i] & 0x0F;

} while(two\_halfs++ < 1);

}

return buf.toString();

}

public static String Sha(String text)

throws NoSuchAlgorithmException, UnsupportedEncodingException {

MessageDigest md;

md = MessageDigest.getInstance("SHA-1");

byte[] sha1hash = new byte[40];

md.update(text.getBytes("iso-8859-1"), 0, text.length());

sha1hash = md.digest();

return convertToHex(sha1hash);

}

public static void main(String[] argv) throws Exception

{

Scanner sc=new Scanner(System.in);

System.out.println(Sha(sc.nextLine()));

}

}

****

**Result**

All the above cipher algorithms are successfully compiled and output verified.

Ex No :

Date :

**Write a program to implement a set of rules combining the secret controls of Bell lapadula model with controls of Biba Model**

**Aim**

To write a program to implement the bell lapadula with biba model

**Algorithm**

The **Bell-Lapadula Model** of protection systems deals with the control of *information flow*. It is a linear non-discretionary model. This model of protection consists of the following components:

* A set of *subjects*, a set of *objects,* and an access control matrix.
* Several ordered security levels. Each subject has a clearance and each object has a classification which attaches it to a security level. Each subject also has a current clearance level which does not exceed its clearance level. Thus a subject can only change to a clearance level below its assigned clearance level.

The set of access rights given to a subject are the following:

* **Read-Only**: The subject can only read the object.
* **Append** : The subject can only write to the object but it cannot read.
* **Execute** : The subject can execute the object but can neither read nor write.
* **Read-Write**: The subject has both read and write permissions to the object.

**Control Attribute**: This is an attribute given to the subject that creates an object. Due to this, the creator of an object can pass any of the above four access rights of that object to any subject. However, it cannot pass the control attribute itself. The creator of an object is also known as the *controller* of that object.

**Restrictions imposed by the Bell-Lapadula Model:**

The following restrictions are imposed by the model:

* **reading down**: A subject has only read access to objects whose security level is below the subject's current clearance level. This prevents a subject from getting access to information available in security levels higher than its current clearance level.
* **writing up**: A subject has append access to objects whose security level is higher than its current clearance level. This prevents a subject from passing information to levels lower than its current level.

The Bell-Lapadula model supplements the access matrix with the above restrictions to provide access control and information flow. For instance, if a subject has read access to an object in the access matrix, it may still not be able to exercise this right if the object is at a security level higher than its clearance level.

**Program**

import java.io.\*;

import java.util.\*;

public class BellBiba

{

public int number;

public String[] user;

public int[] prior,stored;

public String[][] written;

public BellBiba(int number)

{

this.number=number;

user=new String[number];

prior=new int[number];

written=new String[number][10];

stored=new int[number];

}

public void BellLaPadula()

{

System.out.println("Bell LaPadula\nFiles which a user can read");

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

{

System.out.println(user[i]);

for(int j=prior[i]+1;j<number;j++)

{

int l=0;

for(int ii=0;ii<number;ii++)

{

if(prior[ii]==j)

{

l=ii;

break;

}

}

for(int k=0;k<stored[l];k++)

{

System.out.println("-->"+written[l][k]);

}

}

System.out.println();

}

System.out.println("Users and their accepted receipients");

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

{

System.out.println(user[i]);

for(int j=prior[i]-1;j>=0;j--)

{

int l=0;

for(int ii=0;ii<number;ii++)

{

if(prior[ii]==j)

{

l=ii;

break;

}

}

System.out.println("-->"+user[l]);

}

System.out.println();

}

}

public void Biba()

{

System.out.println("Biba Model\nFiles which a user can read");

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

{

System.out.println(user[i]);

for(int j=prior[i]-1;j>=0;j--)

{

int l=0;

for(int ii=0;ii<number;ii++)

{

if(prior[ii]==j)

{

l=ii;

break;

}

}

for(int k=0;k<stored[l];k++)

{

System.out.println("-->"+written[l][k]);

}

}

System.out.println();

}

System.out.println("Users and their accepted receipients");

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

{

System.out.println(user[i]);

for(int j=prior[i]+1;j<number;j++)

{

int l=0;

for(int ii=0;ii<number;ii++)

{

if(prior[ii]==j)

{

l=ii;

break;

}

}

System.out.println("-->"+user[l]);

}

System.out.println();

}

}

public static void main(String[] argv)

{

BellBiba bb=new BellBiba(5);

bb.user[0]="Abc";

bb.user[1]="Def";

bb.user[2]="Ghi";

bb.user[3]="Jkl";

bb.user[4]="Mno";

bb.prior[0]=4;

bb.prior[1]=2;

bb.prior[2]=0;

bb.prior[3]=1;

bb.prior[4]=3;

bb.stored[0]=2;

bb.stored[1]=3;

bb.stored[2]=1;

bb.stored[3]=3;

bb.stored[4]=2;

bb.written[0][0]="Article1.pdf";

bb.written[0][1]="Article2.pdf";

bb.written[1][0]="Article3.docx";

bb.written[1][1]="Article4.pdf";

bb.written[1][2]="Article5.pdf";

bb.written[2][0]="Article6.pdf";

bb.written[3][0]="Article7.docx";

bb.written[3][1]="Article8.rtf";

bb.written[3][2]="Article9.pdf";

bb.written[4][0]="Article10.pdf";

bb.written[4][1]="Article11.pdf";

bb.BellLaPadula();

bb.Biba();

}

**Output**

Ex No :

Date :

**Install the rootkit and study the various options.**

**AIM**

To install the rootkit is study the various options

**ROOTKIT**

1)A rootkit is a compute program designed to provide continued privileged access to a computer while actively hiding its presence.

2)The term rootkit is a connection of two words ’root’ and ‘kit’.

3)Originally Rootkit was a collection of tools that enable administrator level access to a computer (or) network.

4)Root refers to the admin account on unix and linux systems and kit refers to the software components that implements the root.

5)Rootkits are generate associated with malware-such as a trojans, worms, viruses that cancel their existence and action from user in other system process

**ROOTKIT EXAMPLES**

1)Lane davis and steven Dake earliest known rootkit in early 1990’s.

2)NT rootkits-First malicious rootkit targeted at windows os.

3)HackersDefender-early trojan actered/accgmented the os at a very two level of functions calls.

4) machiaveui-first rootkit targeting mac os appeared in os.

5)rootkit creates hidden system and kernel threads

6)Greek-wiretrapping 2004/05

Instrusted installed a rootkit that togethered ericison’s AXE-PBX

7)Zeus-july 2007

Trojan noise that steals banking information

8)Flame-2012 Attacker windows os.

**Steps for installing chrootkit tool in ubuntu through command prompt**

1)sudo apt-get install chkroot kit.

2)sudo chkrootkit

3)To download the latest version of chkrootkit. $wget -c ftp://ftp:panjena.com,br/pub/sy

4)Download the package md5 hash file.

$wget -c <ftp://ftp> pangeia.com.br/pub/seg/pau/chrootkit.md5

5)extract the package

$ tar x2vf chrootkit tar.gz

6)Run the file

**Options in chrootkit**

Usage : -./chrootkit [option][test..]

Options

-h show help and exits

-v show version information and exit

-l show available tests and exits

-d delay

-q quiet mode

-x expert mode

-r dir use div as root directory

-n ship NFS mounted disks

**Result**

Successfully install the rootkit is study the various options

Ex No :

Date :

**Implementing hacking windows login and password**

**Aim**

To hack a windows account through windows login and password.

**Procedure**

1. Go to c:windows:/system32
2. Copy cmd.exe to sethc.exe
3. Rename cmd.exe to sethc.exe
4. Copy the new seche.exe to system32, when windows asks for writing the file. Click yes
5. Logout from guest account and at the user select windows key and press shift key 5 times.
6. Command prompt with full administrator privileges is open.
7. Type NET USER ADMINISTRATOR “XXX” where xxx is the new password and then press enter
8. The command completed successfully and then exit the command prompt.
9. Login into administrator with new password.

**Reason**

1. When shift key is pressed more then 5 times, widows executes a file named sethc.exe located in system32 folder. It does not even check if it’s the same file. Also it runs with the privilege of the current user which is executing the file that is logged user.
2. The file executes even if the guest user is logged off with the system privilege and not the user privilege

**Result**

Thus windows account has been hacked through windows login and password.