**SHA-512**

package weks;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java. util.\*;

class SHA512 {

public static String encryptThisString(String input)

{ try {

MessageDigest md = MessageDigest.getInstance("SHA-512");

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);

}

}

SHA512()

{

Scanner sc=new Scanner(System.in);

System.out.println(" enter text of your choice ");

String s=sc.nextLine();

System.out.println("HashCode Generated by SHA-512 for: ");

String sha3= encryptThisString(s);

System.out.println("\n s:"+sha3);

}

}

public class sha512 {

public static void main(String[] args) {

SHA512 obj=new SHA512();

}

}

**KEY GENERATION**

import java.util.\*;

class keygen{

public static void main(String args[]){

int i;

Scanner sc=new Scanner(System.in);

int key[]=new int[10];

System.out.println("enteer key number :");

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

key[i]=sc.nextInt();

}

int p10[]=new int[10];

p10[0]=key[2];

p10[1]=key[4];

p10[2]=key[1];

p10[3]=key[6];

p10[4]=key[3];

p10[5]=key[9];

p10[6]=key[0];

p10[7]=key[8];

p10[8]=key[7];

p10[9]=key[5];

System.out.println("p10 is ");

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

System.out.print(""+p10[k]);

int L11[]=new int[5];

L11[0]=p10[1];

L11[1]=p10[2];

L11[2]=p10[3];

L11[3]=p10[4];

L11[4]=p10[0];

System.out.println("\n ");

int L12[]=new int[5];

L12[0]=p10[6];

L12[1]=p10[7];

L12[2]=p10[8];

L12[3]=p10[9];

L12[4]=p10[5];

System.out.println("L11=");

for(int j=0;j<L11.length;j++){

System.out.print(""+L11[j]);

}

System.out.println("\n L12=");

for(int j=0;j<L12.length;j++){

System.out.print(""+L12[j]);

}

System.out.println("\n ");

int p81[]=new int[8];

p81[0]=L12[0];

p81[1]=L11[2];

p81[2]=L12[2];

p81[3]=L11[3];

p81[4]=L12[2];

p81[5]=L11[4];

p81[6]=L12[4];

p81[7]=L12[3];

System.out.println("the firstkey is ");

for(int j=0;j<p81.length;j++){

System.out.print(""+p81[j]);

}

System.out.println("\n ");

int L21[]=new int[5];

L21[0]=L11[2];

L21[1]=L11[3];

L21[2]=L11[4];

L21[3]=L11[0];

L21[4]=L11[1];

int L22[]=new int[5];

L22[0]=L12[2];

L22[1]=L12[3];

L22[2]=L12[4];

L22[3]=L12[0];

L22[4]=L12[1];

System.out.print("\n L21==");

for(int j=0;j<L21.length;j++){

System.out.print(L21[j]);

}

System.out.println("\n ");

System.out.println("L22==");

for(int j=0;j<L22.length;j++){

System.out.print(L22[j]);

}

int p82[]=new int[8];

p82[0]=L22[0];

p82[1]=L21[2];

p82[2]=L22[2];

p82[3]=L21[3];

p82[4]=L22[2];

p82[5]=L21[4];

p82[6]=L22[4];

p82[7]=L22[3];

System.out.println(" \n ");

System.out.println("\n the second key is P82==");

for(int j=0;j<p82.length;j++){

System.out.print(""+p82[j]);

}

}}

**ECURVE**

import java.util.Scanner; // week8 problem

public class EllipticCurve {

public static void main(String arg[])

{

float x1,y1,x2,y2,x3,y3,m,a,b;

Scanner sc= new Scanner(System.in);

System.out.println(" enter thee coordiantee of two points");

x1=sc.nextFloat();

y1=sc.nextFloat();

x2=sc.nextFloat();

y2=sc.nextFloat();

a=sc.nextFloat();

if(x1!=x2)// A AND B ARE; DISTINCT

{ m=(y1-y2)/(x1-x2);

x3=m\*m-x1-x2;

y3=-y1+m\*(x1-x3);

System.out.print(x3+" "+y3);

}

else if(y1==-y2)

{System.out.println(" identity");

}

else if(x1==y1 && y1==y2)//C==A OR C==B)

{System.out.println(" c==b or c==a");}

else

{

m=(3\*x1\*x1+a)/(2\*y1);

x3=m\*m- 2 \* x1;

y3= -y1+m\*(x1-x3);

System.out.print(x3+" "+y3);

}}

}

**DHM**

// week 10 Diffie - Hellmann key exchange algorithm

class DHM{

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

private static long power(long a, long b, long p)

{

if (b == 1)

return a;

else

return (((long)Math.pow(a, b)) % p);

}

// the

DHM()

{

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

P = 23;

System.out.println("The value of P:" + P);

G = 9;

System.out.println("The value of G:" + G);

a = 4;

System.out.println("The private key a for A :" + a);

x = power(G, a, P);

b = 3;

System.out.println("The private key X for party B :" + b);

y = power(G, b, P);

ka = power(y, a, P); // Secret key for Alice

kb = power(x, b, P); // Secret key for Bob

System.out.println("Secret key for the party A is:" + ka);

System.out.println("Secret key for the party B is:" + kb);

}

}

public class Main

{

public static void main(String[] args) {

DHM cObj= new DHM();

}

}

**RSA**

import java.math.\*;

import java.util.\*;

class RSA {

RSA()

{ Scanner sc=new Scanner(System.in);

int p, q, n, z, d = 0, e, i;

int msg ;

System.out.println(" enter a message as a number from :1-100");

msg=sc.nextInt();

double c;

BigInteger msgback;

p = 17;

q = 11;

n = p \* q;

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

System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

// e is for public key exponent

if (gcd(e, z) == 1) {

break;

}

}

System.out.println("the value of e = " + e);

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

int x = 1 + (i \* z);

if (x % e == 0) {

d = x / e;

break;

}

}

System.out.println("the value of d = " + d);

c = (Math.pow(msg, e)) % n;

System.out.println("the input message is : " + msg);

System.out.println("Encrypted message is : " + c);

BigInteger N = BigInteger.valueOf(n);

BigInteger C = BigDecimal.valueOf(c).toBigInteger();

msgback = (C.pow(d)).mod(N);

System.out.println("Decrypted message is : "

+ msgback);

}

static int gcd(int e, int z)

{

if (e == 0)

return z;

else

return gcd(z % e, e);

}

}

public class Main

{

public static void main(String[] args) {

RSA rObje=new RSA();

}

}