**Practical VIII**

**Roll No: 08 Date: 24/11/22**

**Aim: Program to implement (i) ElGamal Cryptosystem (ii) Elliptic Curve Cryptosystem.**

**CODE: ElGamal Cryptosystem**

package prac3;

import java.math.\*;

import java.util.\*;

import java.security.\*;

import java.io.\*;

public class ElGamal

{

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

{

BigInteger p, b, c, secretKey;

Random sc = new SecureRandom();

secretKey = new BigInteger("12345678901234567890");

// public key calculation

System.out.println("secretKey = " + secretKey);

p = BigInteger.probablePrime(64, sc);

b = new BigInteger("3");

c = b.modPow(secretKey, p);

System.out.println("p = " + p);

System.out.println("b = " + b);

System.out.println("c = " + c);

// Encryption

System.out.print("Enter your Big Number message -->");

Scanner sca=new Scanner(System.in);

String s = sca.nextLine();

BigInteger X = new BigInteger(s);

BigInteger r = new BigInteger(64, sc);

BigInteger EC = X.multiply(c.modPow(r, p)).mod(p);

BigInteger brmodp = b.modPow(r, p);

System.out.println("Plaintext = " + X);

System.out.println("r = " + r);

System.out.println("EC = " + EC);

System.out.println("b^r mod p = " + brmodp);

// Decryption

BigInteger crmodp = brmodp.modPow(secretKey, p);

BigInteger d = crmodp.modInverse(p);

BigInteger ad = d.multiply(EC).mod(p);

System.out.println("\n\nc^r mod p = " + crmodp);

System.out.println("d = " + d);

System.out.println("Alice decodes: " + ad);

}

}

**OUTPUT**

run:

secretKey = 12345678901234567890

p = 9384985205618657983

b = 3

c = 979090533658577424

Enter your Big Number message -->1873057972239751

Plaintext = 1873057972239751

r = 181396577774081501

EC = 1767538639084685126

b^r mod p = 2935565068434996094

c^r mod p = 1940823507919495599

d = 8088387362150157144

Alice decodes: 1873057972239751

BUILD SUCCESSFUL (total time: 7 minutes 10 seconds)

**CODE: Elliptic Curve Cryptosystem**

package prac3;

import java.math.BigInteger;

import java.security.\*;

import java.security.spec.\*;

import javax.crypto.KeyAgreement;

public class elipticcurve {

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

KeyPairGenerator kpg;

kpg = KeyPairGenerator.getInstance("EC","SunEC");

ECGenParameterSpec ecsp;

ecsp = new ECGenParameterSpec("secp192k1");

kpg.initialize(ecsp);

KeyPair kpU = kpg.genKeyPair();

PrivateKey privKeyU = kpU.getPrivate();

PublicKey pubKeyU = kpU.getPublic();

System.out.println("User U: " + privKeyU.toString());

System.out.println("User U: " + pubKeyU.toString());

KeyPair kpV = kpg.genKeyPair();

PrivateKey privKeyV = kpV.getPrivate();

PublicKey pubKeyV = kpV.getPublic();

System.out.println("User V: " + privKeyV.toString());

System.out.println("User V: " + pubKeyV.toString());

KeyAgreement ecdhU = KeyAgreement.getInstance("ECDH");

ecdhU.init(privKeyU);

ecdhU.doPhase(pubKeyV,true);

KeyAgreement ecdhV = KeyAgreement.getInstance("ECDH");

ecdhV.init(privKeyV);

ecdhV.doPhase(pubKeyU,true);

System.out.println("Secret computed by U: 0x" +

(new BigInteger(1, ecdhU.generateSecret()).toString(16)).toUpperCase());

System.out.println("Secret computed by V: 0x" +

(new BigInteger(1, ecdhV.generateSecret()).toString(16)).toUpperCase());

}}

**OUTPUT**

run:

User U: sun.security.ec.ECPrivateKeyImpl@6d99

User U: Sun EC public key, 192 bits

public x coord: 2659116648645067537841373626533328699445744004967707394010

public y coord: 2925315882912416203978242720322403264388627673308302620533

parameters: secp192k1 (1.3.132.0.31)

User V: sun.security.ec.ECPrivateKeyImpl@ffffc306

User V: Sun EC public key, 192 bits

public x coord: 1771198257990641079745943174206123694031529879505862164726

public y coord: 1115325812363320385883765256693210009229331639836112703868

parameters: secp192k1 (1.3.132.0.31)

Secret computed by U: 0xC1DA624499CE81A908E6711190B6CFD324E161BF1B168834

Secret computed by V: 0xC1DA624499CE81A908E6711190B6CFD324E161BF1B168834

BUILD SUCCESSFUL (total time: 12 seconds)