Name – Jay S. Chatpalliwar

PRN- 21510018

Assignment no 07

import java.io.DataInputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.util.Random;

public class RSA

{

    private BigInteger P;

    private BigInteger Q;

    private BigInteger N;

    private BigInteger PHI;

    private BigInteger e;

    private BigInteger d;

    private int maxLength = 1024;

    private Random R;

    public RSA()

    {

        R = new Random();

        P = BigInteger.probablePrime(maxLength, R);

         Q = BigInteger.probablePrime(maxLength, R);

        N = P.multiply(Q);

       PHI = P.subtract(BigInteger.ONE).multiply(  Q.subtract(BigInteger.ONE));

        e = BigInteger.probablePrime(maxLength / 2, R);

        while (PHI.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(PHI) < 0)

        {

            e.add(BigInteger.ONE);

        }

        d = e.modInverse(PHI);

    }

    public RSA(BigInteger e, BigInteger d, BigInteger N)

    {

        this.e = e;

        this.d = d;

        this.N = N;

    }

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

    {

        RSA rsa = new RSA();

        DataInputStream input = new DataInputStream(System.in);

        String inputString;

        System.out.println("Enter message you wish to send.");

        inputString = input.readLine();

        System.out.println("Encrypting the message: " + inputString);

        System.out.println("The message in bytes is:: "

                + bToS(inputString.getBytes()));

        // encryption

        byte[] cipher = rsa.encryptMessage(inputString.getBytes());

        // decryption

        byte[] plain = rsa.decryptMessage(cipher);

        System.out.println("Decrypting Bytes: " + bToS(plain));

        System.out.println("Plain message is: " + new String(plain));

    }

    private static String bToS(byte[] cipher)

    {

        String temp = "";

        for (byte b : cipher)

        {

            temp += Byte.toString(b);

        }

        return temp;

    }

    // Encrypting the message

    public byte[] encryptMessage(byte[] message)

    {

        return (new BigInteger(message)).modPow(e, N).toByteArray();

    }

    // Decrypting the message

    public byte[] decryptMessage(byte[] message)

    {

        return (new BigInteger(message)).modPow(d, N).toByteArray();

    }

}

