*#----------------------------------------------------------------------------------  
#Rutvik Parmar  
#H-41  
#Date - 7th April 2016  
#Assignement No:8 – Implementation of RSA algorithm   
#----------------------------------------------------------------------------------*

import java.math.BigInteger;

import java.security.SecureRandom;

public class RSA {

private final static BigInteger one = new BigInteger("1");

private final static SecureRandom random = new SecureRandom();

private BigInteger privateKey;

private BigInteger publicKey;

private BigInteger modulus;

// generate an N-bit (roughly) public and private key

RSA(int N) {

BigInteger p = BigInteger.probablePrime(N/2, random);

BigInteger q = BigInteger.probablePrime(N/2, random);

BigInteger phi = (p.subtract(one)).multiply(q.subtract(one));

modulus = p.multiply(q);

publicKey = new BigInteger("65537"); // common value in practice = 2^16 + 1

privateKey = publicKey.modInverse(phi);

}

BigInteger encrypt(BigInteger message) {

return message.modPow(publicKey, modulus);

}

BigInteger decrypt(BigInteger encrypted) {

return encrypted.modPow(privateKey, modulus);

}

public String toString() {

String s = "";

s += "public = " + publicKey + "\n";

s += "private = " + privateKey + "\n";

s += "modulus = " + modulus;

return s;

}

public static void main(String[] args) {

int N = Integer.parseInt(args[0]);

RSA key = new RSA(N);

System.out.println(key);

// create random message, encrypt and decrypt

BigInteger message = new BigInteger(N-1, random);

//// create message by converting string to integer

// String s = "test";

// byte[] bytes = s.getBytes();

// BigInteger message = new BigInteger(bytes);

BigInteger encrypt = key.encrypt(message);

BigInteger decrypt = key.decrypt(encrypt);

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

System.out.println("encrypted = " + encrypt);

System.out.println("decrypted = " + decrypt);

}

}

Ouput –

public = 65537

private = 310963986195833

modulus = 418808671677203

message = 443474853378554

encrypted = 265536089869446

decrypted = 24666181701351

**C code –**

**#include<stdio.h>**

**#include<math.h>**

**int encrypt(int publickey, int n, int message)**

**{**

**int c = 1;**

**int i = 0;**

**for(i = 0; i < publickey; i++)**

**{**

**c = (c\*message)%n;**

**}**

**c = c%n;**

**return c;**

**}**

**int decrypt(int privatekey, int n, int cipher)**

**{**

**int m = 1, i = 0;**

**for(i = 0; i < privatekey; i++)**

**{**

**m = (m\*cipher)%n;**

**}**

**m = m%n;**

**return m;**

**}**

**int main()**

**{**

**int p,q,n,e,d,m;**

**printf("\n\nEnter two relatively prime numbers - ");**

**scanf("%d %d",&p,&q);**

**n = p\*q;**

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

**printf("\nφ(n) - %d",phi);**

**printf("\n\nChoose e such that 1 < e < φ(n) and e and n are coprime - ");**

**scanf("%d",&e);**

**while((d\*e)%(phi)!=1)**

**{**

**d++;**

**}**

**printf("\nPublic Key - (%d,%d)\t",e,n);**

**printf("\nPrivate Key - (%d,%d)\t",d,n);**

**printf("\n\nEnter the plaintext - \t");**

**scanf("%d",&m);**

**int ciphertext = encrypt(e,n,m);**

**printf("\nEncrypted Message - %d\t",ciphertext);**

**int result = decrypt(d,n,ciphertext);**

**printf("\nDecrypted Message - %d\n\n\n", result);**

**return 0;**

**}**