**WEEK-6**

**6.** User A want to communicate to user B but they want to user Asymmetric Key Cryptography by using RSA algorithms send message to each other. encrypt message at sender side and decrypt it at receiver side?

**AIM:** To apply RSA Algorithm using Java

**OBJECTIVE:** To understand the encryption and decryption using Java by RSA encryption algorithm.

**THEORY:**

The **RSA algorithm** is an asymmetric cryptography algorithm; this means that it uses a *public* key and a *private* key (i.e two different, mathematically linked keys). As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.

**RSA in Data Encryption**

There are two broad components when it comes to RSA cryptography, they are:

**Key Generation:** Generating the keys to be used for encrypting and decrypting the data to be exchanged.

**Encryption/Decryption Function:** The steps that need to be run when scrambling and recovering the data.

**Key Generation**

Choose two large prime numbers (p and q)

Calculate n = p\*q and z = (p-1) (q-1)

Choose a number e where 1 < e < z

Calculate d = e-1mod (p-1)(q-1)

Private key pair as (n,d)

Public key pair as (n,e)

**Encryption/Decryption Function**

Once generate the keys, pass the parameters to the functions that calculate your cipher text and plaintext using the respective key.

* If the plaintext is m, cipher text = me mod n.
* If the cipher text is c, plaintext = cd mod n

**ALGORITHM:**

**Step 1** : Choose two prime numbers p and q.

**Step 2** : Calculate n = p\*q

**Step 3** : Calculate  ϕ(n) = (p – 1) \* (q – 1)

**Step 4** : Choose e such that gcd(e , ϕ(n) ) = 1

**Step 5** : Calculate d such that e\*d mod ϕ(n) = 1

**Step 6** : Public Key {e,n} Private Key {d,n}

**Step 7** : Cipher text C = Pe mod n  where P = plaintext

**Step 8** : For Decryption D = Dd mod n where D will give back the plaintext.

**PROGRAM :**

import java.util.\*;

import java.math.\*;

class RSA

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

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

System.out.println("Enter the number to be encrypted and decrypted");

int msg=sc.nextInt();

double c;

BigInteger msgback;

System.out.println("Enter 1st prime number p");

p=sc.nextInt();

System.out.println("Enter 2nd prime number q");

q=sc.nextInt();

n=p\*q;

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

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

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

{

if(gcd(e,z)==1) // e is for public key exponent

{

break;

}

}

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

for(i=0;i<=9;i++)

{

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

if(x%e==0) //d is for private key exponent

{

d=x/e;

break;

}

}

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

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

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

System.out.println(c);

//converting int value of n to BigInteger

BigInteger N = BigInteger.valueOf(n);

//converting float value of c to BigInteger

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

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

System.out.println("Derypted message is : -");

System.out.println(msgback);

}

static int gcd(int e, int z)

{

if(e==0)

return z;

else

return gcd(z%e,e);

}

}

**OUT PUT :**

