Assignment 4
To Study And Understand RSA Algorithm

PRN No. 2018BTECS00212

AIM: To study and understand RSA algorithms.

.

THEORY:

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in 1978 and hence named the **RSA** algorithm. Asymmetric actually means that it works on two different keys i.e. **Public Key** and **Private Key.** As the name describes that the Public Key is given to everyone and the Private key is kept private.

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. A private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Algorithm:-

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −

N=p*q

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: Public key

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

Step 4: Private Key

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −

ed = 1 mod (p-1) (q-1)

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

C = Pe mod n

Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as −

Plaintext = Cd mod n

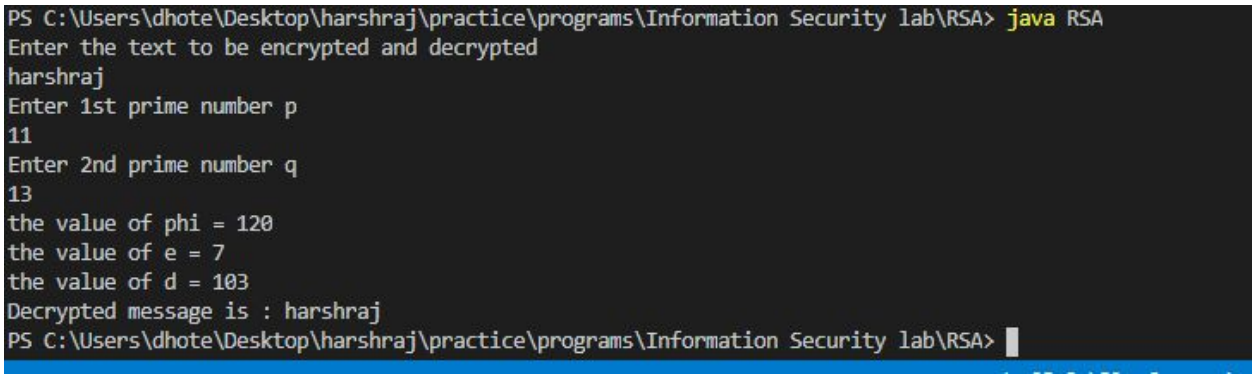
SOURCE CODE:

```
import java.util.*;
import java.math.*;

public class RSA {

    private static final Scanner sc = new Scanner(System.in);

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

    public RSA() {
        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)             {
                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 = x / e;
                break;
            }
        }
        System.out.println("the value of d = " + d);
    }

    private static int gcd(int e, int z) {
        if (e == 0) {
            return z;
        } else {
            return gcd(z % e, e);
        }
    }

    double encrypt(int msg) {
                return (Math.pow(msg, e)) % n;
    }

    double[] encrypt(String msg) {
        int[] charactersAsNumbers = new int[msg.length()];
        for(int i = 0; i < msg.length(); i++) {
            charactersAsNumbers[i] = msg.codePointAt(i);
        }
        System.out.println("Plain text as sequence of numbers: " +
Arrays.toString(charactersAsNumbers));
```

```
        double[] encryptedMsg = new double[msg.length()];
        for(int i = 0; i < charactersAsNumbers.length; i++) {
            encryptedMsg[i] = encrypt(charactersAsNumbers[i]);
        }
        return encryptedMsg;
    }

    BigInteger decrypt(double encrypted) {
                BigInteger N = BigInteger.valueOf(n);
                BigInteger C =
BigDecimal.valueOf(encrypted).toBigInteger();
        return (C.pow(d)).mod(N);
    }

    String decrypt(double[] encrypted) {
        StringBuilder builder = new StringBuilder();
        for(double encryptedCharacter: encrypted) {
            BigInteger decryptedCharacter = decrypt(encryptedCharacter);

builder.append(Character.toChars(decryptedCharacter.intValue()));
        }
        return builder.toString();
    }

    public static void main(String args[]) {
        System.out.println("Enter the text to be encrypted and
decrypted");
        String msg = sc.nextLine();
        RSA rsa = new RSA();

        double[] c = rsa.encrypt(msg);
        System.out.println("Encrypted message is: " + Arrays.toString(c));

        String msgBack = rsa.decrypt(c);
        System.out.println("Decrypted message is: " + msgBack);
    }
}
```

O/P SCREENSHOTS :



Conclusion:

Hence I successfully implemented the RSA algorithm for encryption and decryption.