

**EXP NO:4****DATE:****RSA****AIM:**

To implement an encryption algorithm using RSA.

**ALGORITHM:**

**Step 1:** Generate two distinct prime numbers, p and q.

**Step 2:** Calculate the modulus, n, by multiplying p and q:  $n=p \times q$ .

**Step 3:** Calculate Euler's totient function,  $\phi(n)$ , where

$$\phi(n)=(p-1) \times (q-1)$$

**Step 4:** Choose an integer e such that  $1 < e < \phi(n)$  and e is coprime with  $\phi(n)$ , i.e.,  $\gcd(e, \phi(n)) = 1$ .

**Step 5:** Compute the private key, d, using the modular multiplicative inverse of e modulo  $\phi(n)$ , i.e.,  $d \times e \equiv 1 \pmod{\phi(n)}$ .

**Step 6:** Encrypt the plaintext message, M, using the public key (e, n), where the ciphertext, C, is calculated as  $C \equiv M^e \pmod{n}$ .

**Step 7:** Decrypt the ciphertext, C, using the private key (d, n), where the original message, M, is recovered as  $M \equiv C^d \pmod{n}$ .

**PROGRAM:**

```
import java.io.*;
import java.math.*;
import java.util.*;
public class GFG {
    public static double gcd(double a, double h)
    {
        double temp;
        while (true) {
            temp = a % h;
            if (temp == 0)
```

```
        return h;
        a = h;
        h = temp;
    }
}
public static void main(String[] args)
{
    double p = 9;
    double q = 5;

    double n = p * q;

    double e = 2;
    double phi = (p - 1) * (q - 1);
    while (e < phi) {

        if (gcd(e, phi) == 1)
            break;
        else
            e++;
    }

    int k = 2;
    double d = (1 + (k * phi)) / e;

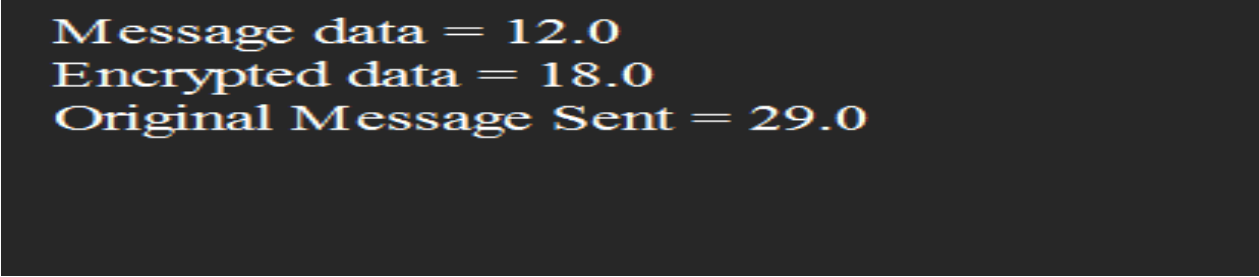
    double msg = 12;

    System.out.println("Message data = " + msg);

    double c = Math.pow(msg, e);
    c = c % n;
    System.out.println("Encrypted data = " + c);
}
```

```
        double m = Math.pow(c, d);  
        m = m % n;  
        System.out.println("Original Message Sent = " + m);  
    }  
}
```

**OUTPUT:**



```
Message data = 12.0  
Encrypted data = 18.0  
Original Message Sent = 29.0
```

**RESULT:**