

**EXP NO:4**

**DATE:**

## **RSA**

**Aim:** To implement an encryption algorithm using Rsa.

### **Algorithm:**

- Step 1: Select two large prime numbers, p and q.
- Step 2: Calculate the modulus,  $n = p * q$ .
- Step 3: Compute Euler's totient function,  $\phi(n) = (p - 1) * (q - 1)$ .
- Step 4: Choose a public exponent, e, such that  $1 < e < \phi(n)$  and  $\gcd(e, \phi(n)) = 1$ .
- Step 5: Compute the private exponent, d, such that  $(d * e) \bmod \phi(n) = 1$ .
- Step 6: Convert the plaintext message into a numerical representation, usually using ASCII values or Unicode.
- Step 7: Encrypt the message by computing ciphertext, c, using the formula  $c = (\text{msg}^e) \bmod n$ .
- Step 8: Print the encrypted data.
- Step 9: Decrypt the ciphertext by computing the original message, m, using the formula  $m = (c^d) \bmod n$ .
- Step 10: Print the original message.
- Step 11: Return 0 for successful execution and program termination.

### **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;

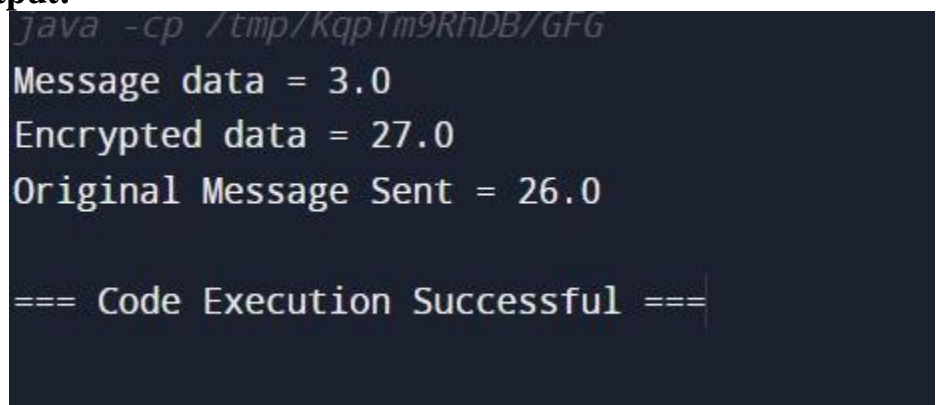
    double msg = 12;

    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:**

A screenshot of a terminal window with a dark background. The first line shows a Java command: `java -cp /tmp/kqp1m9RhDB/GFG`. The subsequent lines show the program's output: `Message data = 3.0`, `Encrypted data = 27.0`, and `Original Message Sent = 26.0`. The final line is a separator: `=== Code Execution Successful ===`.

```
java -cp /tmp/kqp1m9RhDB/GFG  
Message data = 3.0  
Encrypted data = 27.0  
Original Message Sent = 26.0  
  
=== Code Execution Successful ===
```

**Result:**