

### Experiment no.5: Implement RSA Algorithm

**Learning Objective:** Student should be able to implement RSA algorithm

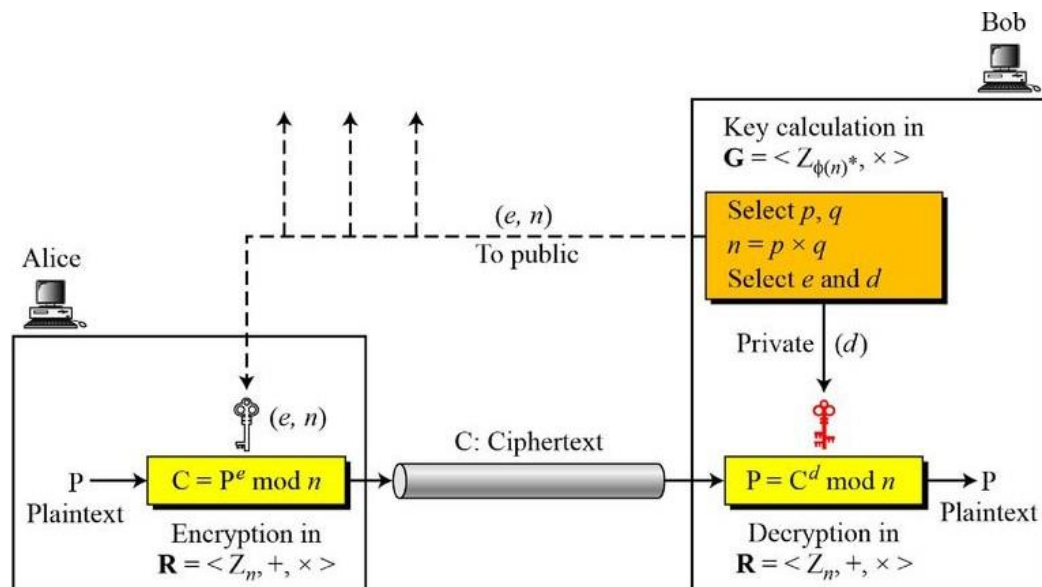
**Tools:** C/C++/Java/Python or any computational software

#### Theory:

The most common public-key algorithm is the RSA cryptosystem, named for its inventors (Rivest, Shamir, and Adleman).

RSA uses two exponents,  $e$  and  $d$ , where  $e$  is public and  $d$  is private. Suppose  $P$  is the plaintext and  $C$  is the ciphertext. Alice uses  $C = P^e \bmod n$  to create ciphertext  $C$  from plaintext  $P$ ; Bob uses  $P = C^d \bmod n$  to retrieve the plaintext sent by Alice. The modulus  $n$ , a very large number, is created during the key generation process, that will be discussed later. Encryption and decryption use modular exponentiation. Modular exponentiation is feasible in polynomial time using the fast exponentiation algorithm. However, modular logarithm is as hard as factoring the modulus, for which there is no polynomial algorithm yet. This means that Alice can encrypt in polynomial time ( $e$  is public), Bob also can decrypt in polynomial time (because he knows  $d$ ), but Eve cannot decrypt because she would have to calculate the  $e$ th root of  $C$  using modular arithmetic.

#### RSA Encryption, Decryption and Key generation:



### RSA Key Generation, encryption and decryption algorithm:

- ▶ Select at random two large prime numbers  $p, q$ .
- ▶ Compute  $n = pq$  and  $\phi(n) = (p - 1)(q - 1)$ .
- ▶ Select a small integer  $e$  such that  $\gcd(e, \phi(n)) = 1$ .
- ▶ Compute  $d = e^{-1} \mod \phi(n)$ .
- ▶ Publish the pair  $(n, e)$  as your RSA Public Key.
- ▶ Keep secret  $d$ , your RSA Private Key. Also keep  $p, q, \phi(n)$  secret.

To encrypt a message  $M$  we find  $C = M^e \mod n$ .

To decrypt a cyphertext  $C$  we find  $M = C^d \mod n$ .

#### Example:

Bob chooses 7 and 11 as  $p$  and  $q$  and calculates  $n = 7 \times 11 = 77$ . The value of  $\phi(n) = (7 - 1)(11 - 1)$  or 60. Now he chooses two exponents,  $e$  and  $d$ , from  $Z_{60}^*$ . If he chooses  $e$  to be 13, then  $d$  is 37. Note that  $e \times d \mod 60 = 1$  (they are inverses of each other). Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5.

Plaintext: 5

$$C = 5^{13} = 26 \mod 77$$

Ciphertext: 26

Bob receives the ciphertext 26 and uses the private key 37 to decipher the ciphertext:

Ciphertext: 26

$$P = 26^{37} = 5 \mod 77$$

Plaintext: 5

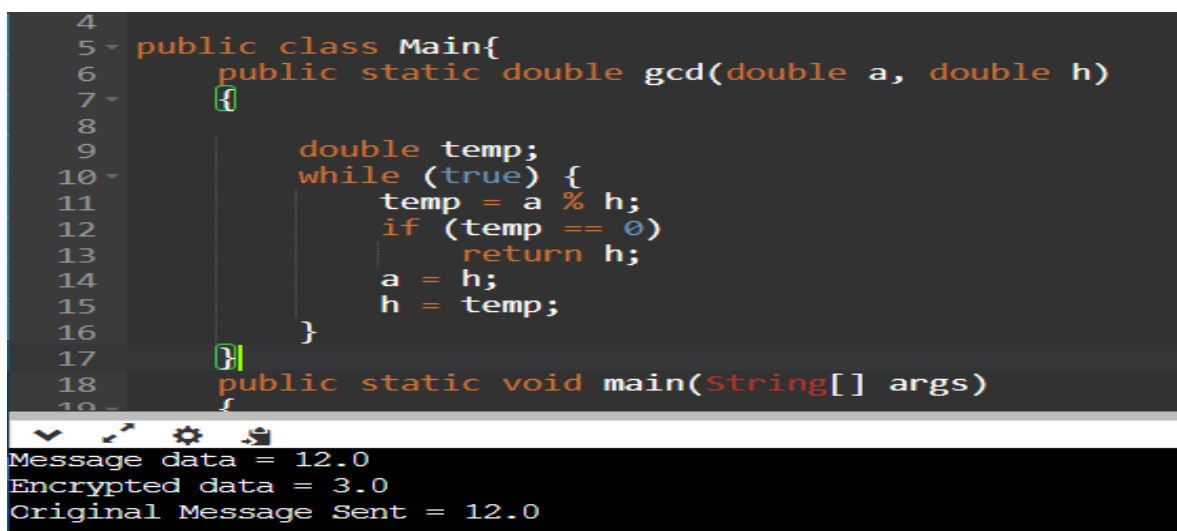
The plaintext 5 sent by Alice is received as plaintext 5 by Bob.

#### Source Code:

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

```
public static void main(String[] args)
{
    double p = 3;
    double q = 7;
    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:



```
4
5 public class Main{
6     public static double gcd(double a, double h)
7     {
8
9         double temp;
10        while (true) {
11            temp = a % h;
12            if (temp == 0)
13                return h;
14            a = h;
15            h = temp;
16        }
17    }
18    public static void main(String[] args)
19    {
20        Message data = 12.0
21        Encrypted data = 3.0
22        Original Message Sent = 12.0
```

### Applications:

1. **Digital signatures:** A digital signature is a technique that lets the recipient of a message verify its authenticity, integrity and non-repudiation. It proves that the message has not been altered in transit.
2. **Secure key exchange:** Another use case of RSA is to have a secure key exchange between two parties who have not previously shared a secret key. The two parties involved generate a public-private key pair using the RSA algorithm.

**Result and Discussion:** The implementation of the RSA algorithm yielded promising results, demonstrating its effectiveness in encrypting and decrypting messages securely. Through encryption, plaintext messages were successfully transformed into cipher text using the public key, ensuring confidentiality. Decryption using the private key efficiently restored the original plaintext, confirming the algorithm's reliability in maintaining data integrity. Additionally, the experiment highlighted the importance of key size and generation in enhancing security against potential attacks, reaffirming RSA's significance in modern cryptography.

**Learning Outcomes:** The student should have the ability to implement RSA Algorithm

LO1: To describe & understand about RSA Algorithm

LO2: To implement RSA algorithm

**Course Outcomes:** Upon completion of the course students will be able to understand & implement RSA algorithm.

**Conclusion:** In conclusion, the implementation of the RSA algorithm offers several advantages, including robust security through the use of public and private keys, facilitating secure communication over insecure channels. Additionally, RSA provides a standardized method for encryption and digital signatures, supporting a wide range of applications in data security. However, its computational complexity can be a significant drawback, especially for large-scale deployments, potentially impacting performance in resource-constrained environments. Despite this limitation, the RSA algorithm remains a cornerstone of modern cryptography, balancing security and practicality in safeguarding sensitive information.



**TCET**  
**BE COMPUTER SCIENCE & ENGINEERING (CYBER SECURITY)**  
Choice Based Credit Grading System (CBCGS)  
Under TCET Autonomy



**Name:** Aryan Tiwari

**Class:** SE – CSE

**Rollno:** 57

**For Faculty Use**

Correction Parameter	Formative Assessment [40%]	Timely completion of Practical [ 40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				