Government College of Engineering, Jalgaon
(An Autonomous Institute of Govt. of Maharashtra)

Name: Pratiksha Rajendra Badgujar	**PRN:** 2241003
Class: L.Y **Semester:** VII	**Batch:** B1
Date of Performance: _____	**Date of Completion:** _____
Subject: CO407U CNSL	**Subject Teacher:** Ms. Shrutika Mahajan

Practical – 8

Aim: Implement RSA Algorithm

Theory:

Introduction:-

RSA stands for Rivest-Shamir-Adleman. RSA algorithm is an asymmetric cryptography algorithm. Asymmetric means that it works on two different keys

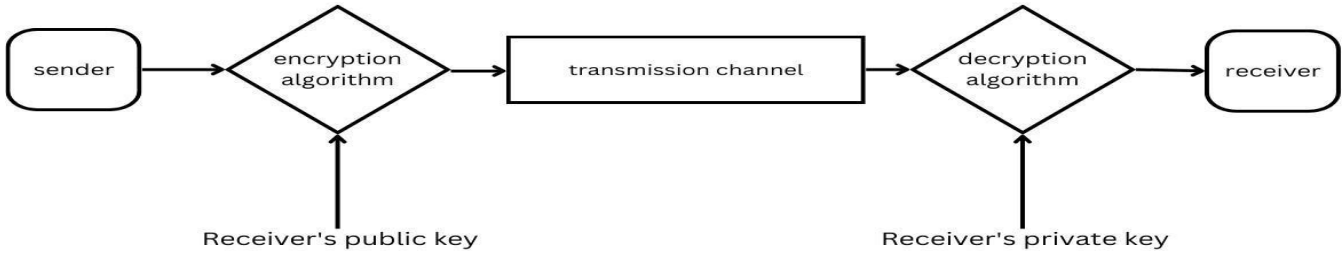
1 Public key
2 Private key

Public key for encryption and a Private key for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using user's public key.

But only the user can decrypt the message using his private key.

Asymmetric encryption, commonly referred to as public-key cryptography, uses two distinct keys for encryption and decryption. The public key, which is extensively used to encrypt data and is known to all, is one type of key. The private key, on the other hand, is kept private i.e., only the receiver knows it and is used to decode data.

Both, the public key and the private key should be available at both the sender's end and the receiver's end for the asymmetric encryption to succeed.



Asymmetric Encryption

The encryption algorithm receives the sender's plain text message, encrypts it using the recipient's public key, and generates a cipher. The recipient then receives this cipher via a transmission or communication channel. The decryption process on the receiver's end uses the decryption algorithm and the receiver's private key to recover the original plain text message

RSA Encryption and Decryption:-

Steps:-

1. **Key Generation**
   - RSA requires two keys a public key and a private key.
   - The public key is used for encryption, while the private key is used for decryption.
2. **Encryption**
   - Use the public key to encrypt the message.
3. **Decryption**
   - Use the private key to decrypt the encrypted message.

ALGORITHM :-

1. Select 2 prime numbers, preferably large, p and q.
2. Calculate n = p*q.
3. Calculate phi(n) = (p-1)*(q-1)
4. Choose a value of e such that 1<e<phi(n) and gcd(phi(n), e) = 1.
5. Calculate d such that d = (e^-1) mod phi(n).

Here the public key is {e, n} and private key is {d, n}. If M is the plain text then the cipher text C = (M^e) mod n. This is how data is encrypted in RSA algorithm. Similarly, for decryption, the plain text M = (C^d) mod n.

Code:

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

public class RSAUserDefined {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Step 1: Input p and q (must be prime)
      System.out.print("Enter first prime number (p): ");
      BigInteger p = sc.nextBigInteger();

      System.out.print("Enter second prime number (q): ");
```

```
BigInteger q = sc.nextBigInteger();

// Step 2: Compute n = p * q
BigInteger n = p.multiply(q);

// Step 3: Compute φ(n) = (p-1)*(q-1)
BigInteger phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

// Step 4: Choose e such that 1 < e < φ(n) and gcd(e, φ(n)) = 1
BigInteger e = new BigInteger("2");
while (e.compareTo(phi) < 0) {
    if (e.gcd(phi).equals(BigInteger.ONE))
        break;
    e = e.add(BigInteger.ONE);
}

// Step 5: Compute d, the modular inverse of e mod φ(n)
BigInteger d = e.modInverse(phi);

System.out.println("\n--- RSA Key Generation ---");
System.out.println("p = " + p);
System.out.println("q = " + q);
System.out.println("n = p * q = " + n);
System.out.println("φ(n) = (p-1)*(q-1) = " + phi);
System.out.println("Public key (e, n): (" + e + ", " + n + ")");
System.out.println("Private key (d, n): (" + d + ", " + n + ")");
```

```
        // Step 6: Encryption
        System.out.print("\nEnter plaintext (numeric): ");
        BigInteger msg = sc.nextBigInteger();

        BigInteger ciphertext = msg.modPow(e, n);
        System.out.println("Ciphertext (C) = M^e mod n = " + ciphertext);

        // Step 7: Decryption
        BigInteger plaintext = ciphertext.modPow(d, n);
        System.out.println("Plaintext (Decrypted M) = C^d mod n = " + plaintext);

        sc.close();
    }
}
```

Output:

```
PS D:\Me\Computer\Study Material\CNS Practi\Program> java  RSAUserDefined
Enter first prime number (p):
11
Enter second prime number (q): 13

--- RSA Key Generation ---
p = 11
q = 13
n = p * q = 143
??(n) = (p-1)*(q-1) = 120
Public key (e, n): (7, 143)
Private key (d, n): (103, 143)

Enter plaintext (numeric): 9
Ciphertext (C) = M^e mod n = 48
Plaintext (Decrypted M) = C^d mod n = 9
```

Conclusion:

The RSA algorithm provides a secure method for encrypting and decrypting data using public and private keys. It ensures data confidentiality and integrity through strong mathematical foundations in prime factorization.

Course Teacher
Ms. Shrutika Mahajan