BRACT's
Vishwakarma Institute of Information Technology
Department of Computer Engineering(Software Engineering)

Submitted By :

Roll No.	Name	PRN No.
3101070	Pranita Narayan Kute	22420110

ASSIGNMENT NO. 5

1. **Title** - Implement asymmetric key encryption using RSA
2. **Aim -** To implement Asymmetric Key Encryption using the RSA algorithm for secure data transmission between sender and receiver.
3. **Objective**
   a. To understand the concept of public-key cryptography.
   b. To generate public and private keys using the RSA algorithm.
   c. To perform encryption using the public key and decryption using the private key.
   d. To demonstrate the security principle that even if the public key is known, the private key cannot be easily derived.
4. **Pre-requisite Knowledge**

Before implementing RSA, you should be familiar with:

a. Concepts of encryption and decryption.
b. Number theory basics: prime numbers, modular arithmetic, Euler's totient function.
c. Python/C++ programming (for practical implementation).
d. Understanding of how public and private keys work in asymmetric encryption.

5. **Theory**

- **Introduction:**

  RSA(Rivest-Shamir-Adleman) Algorithm is an asymmetric or **public-key cryptography** algorithm which means it works on two different keys: Public Key and

Private Key. The Public Key is used for encryption and is known to everyone, while the Private Key is used for decryption and must be kept secret by the receiver. RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.

Example of Asymmetric Cryptography:

If Person A wants to send a message securely to Person B:

1. Person A encrypts the message using Person B's Public Key.
2. Person B decrypts the message using their Private Key.

- **RSA Algorithm Steps**

1. Select two large prime numbers, p and q.
2. Compute n = p * q.
3. Compute Euler's totient function: φ(n) = (p - 1) * (q - 1).
4. Choose an integer e (public key exponent) such that:
   - 1 < e < φ(n)
   - gcd(e, φ(n)) = 1 (i.e., e and φ(n) are coprime)
5. Compute the private key d such that:
   - (d * e) % φ(n) = 1
6. The public key is (e, n) and the private key is (d, n).
7. Encryption:
   - Ciphertext C = (M^e) % n where M is the plaintext message.
8. Decryption:
   - Plaintext M = (C^d) % n

CODE

```
#include <bits/stdc++.h>

using namespace std;

// Function to find GCD

int gcd(int a, int b) {

   if (b == 0)

      return a;

   return gcd(b, a % b);

}
```

```
// Function to find Multiplicative Inverse of e mod phi
int modInverse(int e, int phi) {
    for (int d = 1; d < phi; d++) {
        if ((d * e) % phi == 1)
            return d;
    }
    return -1;
}
// Function to perform modular exponentiation
long long power(long long base, long long exp, long long mod) {
    long long result = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;
        exp = exp >> 1; // divide by 2
        base = (base * base) % mod;
    }
    return result;
}
int main() {
    // Step 1: Select two prime numbers
    int p = 17, q = 11;
    int n = p * q;
```

```
    int phi = (p - 1) * (q - 1);

    // Step 2: Choose e such that 1 < e < phi and gcd(e, phi) = 1

    int e;

    for (e = 2; e < phi; e++) {

        if (gcd(e, phi) == 1)

            break;

    }

    // Step 3: Find d, the multiplicative inverse of e mod phi

    int d = modInverse(e, phi);

    cout << "Public Key: (" << e << ", " << n << ")\n";

    cout << "Private Key: (" << d << ", " << n << ")\n";

    // Step 4: Input message

    int msg;

    cout << "\nEnter message (integer < " << n << "): ";

    cin >> msg;

    // Step 5: Encryption -> C = (M^e) mod n

    long long cipher = power(msg, e, n);

    cout << "Encrypted Message: " << cipher << endl;

    // Step 6: Decryption -> M = (C^d) mod n

    long long decrypted = power(cipher, d, n);

    cout << "Decrypted Message: " << decrypted << endl;

    return 0;

}
```

6. Conclusion

- RSA demonstrates asymmetric encryption, where two different keys are used for encryption and decryption.
- It ensures confidentiality, authentication, and non-repudiation in communication systems.
- The security of RSA relies on the difficulty of factoring large prime numbers.
- This implementation provides the foundation for real-world cryptographic systems, such as digital signatures, SSL/TLS, and secure email.

7. Resources

Sr. No.	Source	Link
1.	Tutorialspoint	https://www.tutorialspoint.com/cryptography/md5_algorithm.htm
2.	GeeksForGeeks	https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography/