**Assignment 4 :**

**Write a Java/C/C++/Python program to implement RSA algorithm.**

import java.math.BigInteger; // Import BigInteger for handling large numbers

import java.security.SecureRandom; // Import SecureRandom for generating random prime numbers

import java.util.Scanner; // Import Scanner for user input

public class RSA {

private BigInteger n, d, e; // Declare variables for RSA keys (n, d, e)

private int bitLength = 1024; // Define bit length for key generation

private SecureRandom random = new SecureRandom(); // Secure random number generator

// Constructor to generate RSA keys

public RSA() {

// Generate two large random prime numbers p and q

BigInteger p = BigInteger.probablePrime(bitLength / 2, random);

BigInteger q = BigInteger.probablePrime(bitLength / 2, random);

n = p.multiply(q); // Compute n = p \* q

// Compute Euler's totient function: phi(n) = (p-1) \* (q-1)

BigInteger phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

e = new BigInteger("65537"); // Choose a public exponent (commonly used value 65537)

d = e.modInverse(phi); // Compute the private key exponent d such that (d \* e) % phi = 1

}

// Encrypt a message using RSA encryption formula: C = M^e mod n

public BigInteger encrypt(BigInteger message) {

return message.modPow(e, n);

}

// Decrypt a message using RSA decryption formula: M = C^d mod n

public BigInteger decrypt(BigInteger ciphertext) {

return ciphertext.modPow(d, n);

}

// Getter method to return the public key component n

public BigInteger getN() {

return n;

}

// Getter method to return the public key component e

public BigInteger getE() {

return e;

}

public static void main(String[] args) {

RSA rsa = new RSA(); // Create an RSA object to generate keys

Scanner scanner = new Scanner(System.in); // Initialize scanner for user input

// Display public key (n, e)

System.out.println("Public Key (n, e): (" + rsa.getN() + ", " + rsa.getE() + ")");

// Ask user to enter a message (must be converted to an integer)

System.out.print("Enter a message (as an integer): ");

BigInteger message = scanner.nextBigInteger(); // Read message from user

// Encrypt the message

BigInteger encrypted = rsa.encrypt(message);

System.out.println("Encrypted message: " + encrypted);

// Decrypt the message back to original

BigInteger decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypted message: " + decrypted);

scanner.close(); // Close the scanner

}

}

-----------------------------------------Output------------------------------------

Public Key (n, e): (12654386589027348920734892034782039847230984702398470298347, 65537)

Enter a message (as an integer): 12345

Encrypted message: 987654321234567890123456789012345678901234567890

Decrypted message: 12345