Lab 4

RSA Algorithm: RSA (Rivest–Shamir–Adleman) is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of the keys can be given to anyone. The other key must be kept private. The algorithm is based on the fact that finding the factors of a large composite number is difficult.

- **Program**

- **Implement RSA algorithm. 1. Do Key generation 2. Encryption 3. Decryption. Use only Multiply and Square to do Modular Exponentiation. One can use Miller Rabin for Primality Testing (Optional but recommended) . Use large prime numbers due to security concerns.**

- **Code:**

```
import java.util.Scanner;
import java.util.Random;
import java.math.BigInteger;

public class RSAAlgorithm
{
   public long[] calculateGCD(long r1,long r2)
   {
      long arr[] = new long[2];
      long s1 = 0, s2 = 1, t1 = 1, t2 = 0, temp;
      while (r2 != 0)
      {
         long q = r1 / r2;
         long r = r1 % r2;

         r1 = r2;
         r2 = r;
```

```
            temp = s1;
            s1 = t1 - q * s1;
            t1 = temp;

            temp = s2;
            s2 = t2 - q * s2;
            t2 = temp;
        }
        arr[0] = r1;
        arr[1] = t1;
        return arr;
    }

    public long modMulInverse(long a,long mod)
    {
        long arr[] = calculateGCD(a, mod);
        long inverse = 0;

        if(arr[0] != 1)
        {
            System.out.println("Inverse of " + a +" is not possible!!! with
respect to Z"+mod);
            System.exit(0);
        }
        else
        {
            if(arr[1] < 0)
            {
                inverse = arr[1] + mod;
            }
            else
            {
                inverse = arr[1];
            }
        }
        return inverse;
    }
```

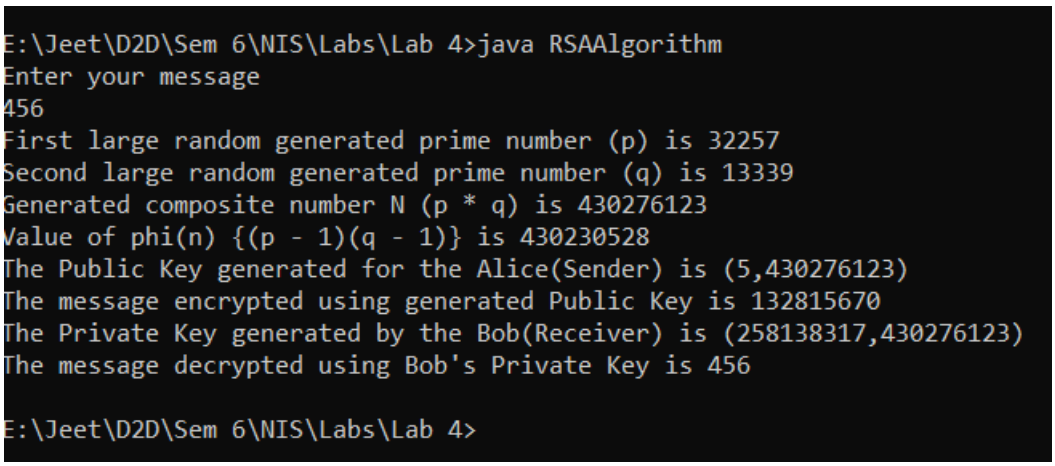
```
   public long modularExponentiation(long base,long exponent,long
modulo)
   {
      long result = 1;

      base = base % modulo;

      while(exponent > 0)
      {
         if((exponent & 1) == 1)
            result = (result * base) % modulo;

         exponent >>= 1;
         base = (base * base) % modulo;
      }
      return result;
   }
   public boolean miilerRabinPrimalityTest(long d,long largeNum)
   {
      long a = 2 + (int)(Math.random() % (largeNum - 4));

      long ans = modularExponentiation(a, d, largeNum);

      if(ans == 1 || ans == largeNum - 1)
         return true;

      while(d != largeNum - 1)
      {
         ans = (ans * ans) % largeNum;
         d *= 2;

         if (ans == 1)
            return false;
         if (ans == largeNum - 1)
            return true;
      }
      return false;
   }
   public boolean isPrimeNumber(long largeNum,int iterations)
   {
```

```
    if(largeNum <= 1)
        return false;
    if(largeNum <= 3)
        return true;

    long d = largeNum - 1;

    while(d % 2 == 0)
        d /= 2;

    for(int i = 0; i <= iterations; i++)
    {
        if(!miilerRabinPrimalityTest(d, largeNum))
            return false;
    }
    return true;
}

public long encryption(long message,long exponent,long n)
{
    long cipher = modularExponentiation(message, exponent, n);
    return cipher;
}

public long decryption(long cipher,long d,long n)
{
    long message = modularExponentiation(cipher, d, n);
    return message;
}
public static void main(String []args)
{
    RSAAlgorithm rsa = new RSAAlgorithm();
    Scanner sc = new Scanner(System.in);
    Random random = new Random();

    int  max = 2,max1 = 2,iterations = 5,exponent;
    long message;

    byte[] bytes = new byte[max];
    byte[] bytes1 = new byte[max1];
```

```
        BigInteger largeNum1;
        BigInteger largeNum2;

        long longVal1,longVal2;


        System.out.println("Enter your message");
        message = sc.nextLong();

        while(true)
        {
           random.nextBytes(bytes);
           largeNum1 = new BigInteger(bytes);
           longVal1 = largeNum1.longValue();


           if(rsa.isPrimeNumber(longVal1, iterations))
              break;
           else
              rsa.isPrimeNumber(longVal1, iterations);
        }
        System.out.println("First large random generated prime number (p) is
"+largeNum1.abs());

        while(true)
        {
           random.nextBytes(bytes1);
           largeNum2 = new BigInteger(bytes1);
           longVal2 = largeNum2.longValue();

           if(rsa.isPrimeNumber(longVal2, iterations))
              break;
           else
              rsa.isPrimeNumber(longVal2, iterations);
        }
        System.out.println("Second large random generated prime number (q)
is "+largeNum2.abs());

        long n = longVal1 * longVal2;
```

```
        long phin = (longVal1 - 1) * (longVal2 - 1);

        System.out.println("Generated composite number N (p * q) is "+n);
        System.out.println("Value of phi(n) {(p - 1)(q - 1)} is "+phin);

        for(exponent = 2; exponent < phin; exponent += 1)
        {
           long arr[] = rsa.calculateGCD(exponent, phin);

           if(arr[0] == 1)
              break;
        }

        System.out.println("The Public Key generated for the Alice(Sender) is
("+exponent+","+n+")");

        long cipher = rsa.encryption(message, exponent, n);
        System.out.println("The message encrypted using generated Public
Key is "+cipher);

        long d = rsa.modMulInverse(exponent, phin);

        System.out.println("The Private Key generated by the Bob(Receiver)
is ("+d+","+n+")");

        System.out.println("The message decrypted using Bob's Private Key
is "+rsa.decryption(cipher, d, n));
    }
}
```

- **Output:**


E:\Jeet\D2D\Sem 6\NIS\Labs\Lab 4>java RSAAlgorithm
Enter your message
456
First large random generated prime number (p) is 32257
Second large random generated prime number (q) is 13339
Generated composite number N (p * q) is 430276123
Value of phi(n) {(p - 1)(q - 1)} is 430230528
The Public Key generated for the Alice(Sender) is (5,430276123)
The message encrypted using generated Public Key is 132815670
The Private Key generated by the Bob(Receiver) is (258138317,430276123)
The message decrypted using Bob's Private Key is 456
E:\Jeet\D2D\Sem 6\NIS\Labs\Lab 4>