

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Системи безпеки програм і даних» на тему
«Розробка програми керування ключами шифрування за
схемою RSA»

ВИКОНАВ:
студент III курсу ФІОТ
групи ІВ-91
Красновський Олексій

ПЕРЕВІРИВ:
аспірант
Іваніщев Б.В.

Київ – 2021

Завдання:

Розробка програми керування ключами шифрування за схемою RSA або El-Gamal (за варіантами). Програма реалізує ту чи іншу схему розрахунку ключів шифрування, обов'язково використовуючи для цього розширений або класичний алгоритми Евкліда, і формуючи відкриті і закриті (секретні) ключі шифрування, які відразу ж використовуються для шифрування повідомлень. Програма реалізує два режими шифрування і дешифрування повідомлень з перевіркою достовірності результатів на основі аналізу цифрового паспорта-сертифіката відкритих ключів.

Аналіз засобів рішення задачі:

При аналізі завдання лабораторної роботи 5, проблем не виникло. При генерації великих простих чисел була використана реалізація тесту Рабіна з попередньої роботи.

Лістинг програми:

RabinPrimalityTestUtils.java

```
package util;

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

public class RabinPrimalityTestUtils {

    private static final int[] smallPrimeNumbers = {2, 3, 5, 7, 11, 13};

    public static BigInteger getPrimeNumber(long PMax) {
        BigInteger a = BigInteger.valueOf(getRandomSmallPrimeNumber());
        BigInteger K = BigInteger.valueOf(Math.round(customLog(a.doubleValue(), (double) PMax / 2)));
        BigInteger P1 = pow(a, K).multiply(BigInteger.TWO).add(BigInteger.ONE);
        BigInteger P2 = pow(a, K).multiply(BigInteger.TWO).subtract(BigInteger.ONE);
        while (true) {
            if (isPrime(a, P1)) return P1;
        }
    }
}
```

```

        if (isPrime(a, P2)) return P2;
        P1 = P1.add(BigInteger.TWO);
        P2 = P2.subtract(BigInteger.TWO);
    }
}

private static int getRandomSmallPrimeNumber() {
    return smallPrimeNumbers[new Random().nextInt(smallPrimeNumbers.length)];
}

public static double customLog(double base, double logNumber) {
    return Math.log(logNumber) / Math.log(base);
}

private static boolean isPrime(BigInteger a, BigInteger P) {
    BigInteger powResult = pow(a, P.subtract(BigInteger.ONE));
    BigInteger result = powResult.mod(P);
    return result.equals(BigInteger.ONE);
}

public static BigInteger pow(BigInteger base, BigInteger exponent) {
    BigInteger result = BigInteger.ONE;
    while (exponent.signum() > 0) {
        if (exponent.testBit(0)) result = result.multiply(base);
        base = base.multiply(base);
        exponent = exponent.shiftRight(1);
    }
    return result;
}
}

```

RSAUtils.java

```

package util;

import static util.RabinPrimalityTestUtils.customLog;

import java.math.BigDecimal;

```

```
import java.math.BigInteger;
```

```
public class RSAUtils {
```

```
    public static BigInteger getPublicKey(BigInteger P, BigInteger FE) {
```

```
        BigInteger start = BigInteger.valueOf(Math.round(customLog(2.0, P.doubleValue())));
```

```
        BigInteger Ko = start.add(BigInteger.ONE);
```

```
        while (true) {
```

```
            if (gcd(Ko, FE).equals(BigInteger.ONE)) {
```

```
                return Ko;
```

```
            }
```

```
            Ko = Ko.add(BigInteger.ONE);
```

```
        }
```

```
    }
```

```
    public static BigInteger getSecretKey(BigInteger Ko, BigInteger Fe) {
```

```
        long z = 1;
```

```
        while (true) {
```

```
            BigDecimal[] tmp = new
```

```
BigDecimal(Fe.multiply(BigInteger.valueOf(z)).add(BigInteger.ONE)).divideAndRemainder(new  
BigDecimal(Ko));
```

```
            if (tmp[1].equals(BigDecimal.ZERO)) {
```

```
                return tmp[0].toBigInteger();
```

```
            }
```

```
            ++z;
```

```
        }
```

```
    }
```

```
    public static BigInteger[] evaluateKeys() {
```

```
        BigInteger[] result = new BigInteger[3];
```

```
        BigInteger p = RabinPrimalityTestUtils.getPrimeNumber(30);
```

```
        BigInteger q = RabinPrimalityTestUtils.getPrimeNumber(30);
```

```
        BigInteger P = p.multiply(q);
```

```
        BigInteger FE = eulerFunction(p, q);
```

```
        result[0] = getPublicKey(P, FE);
```

```
        result[1] = getSecretKey(result[0], FE);
```

```
        result[2] = P;
```

```

    return result;
}

private static BigInteger eulerFunction(BigInteger p, BigInteger q) {
    return p
        .subtract(BigInteger.ONE)
        .multiply(q.subtract(BigInteger.ONE));
}

private static BigInteger gcd(BigInteger a, BigInteger b) {
    if (a.equals(BigInteger.ZERO))
        return b;
    return gcd(b.mod(a), a);
}

}

```

Main.java

```

import static util.RabinPrimalityTestUtils.pow;

import java.math.BigInteger;
import util.RSAUtils;
import util.RabinPrimalityTestUtils;

public class Main {

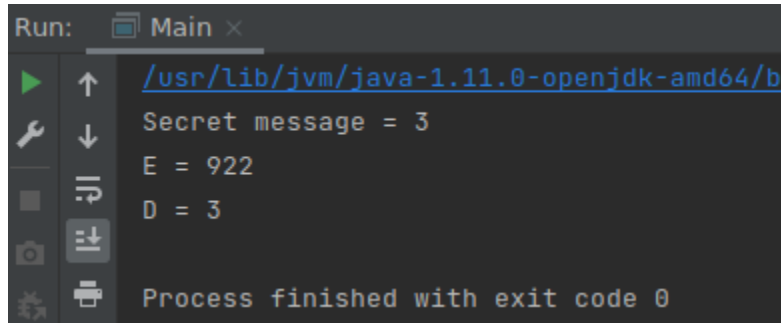
    public static void main(String[] args) {

        // lab 5
        // 0 - public key; 1 - secret key; 2 - P
        BigInteger[] keys = RSAUtils.evaluateKeys();
        BigInteger message = BigInteger.valueOf(3);
        System.out.println("Secret message = " + message);
        BigInteger E = pow(message, keys[0]).mod(keys[2]);
        System.out.println("E = " + E);
        BigInteger D = pow(E, keys[1]).mod(keys[2]);
    }
}

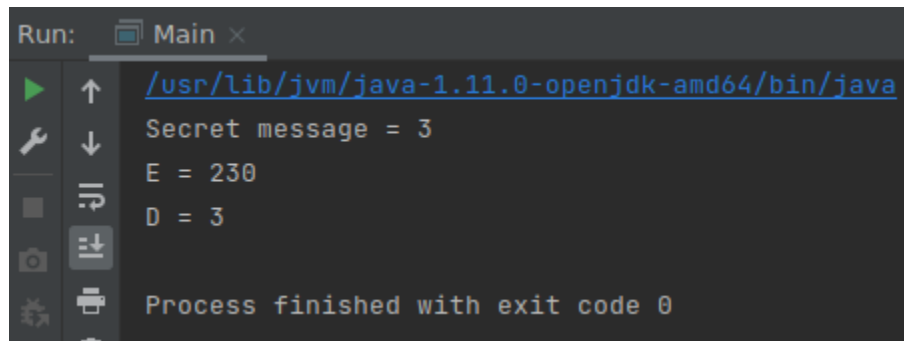
```

```
System.out.println("D = " + D);  
}  
}
```

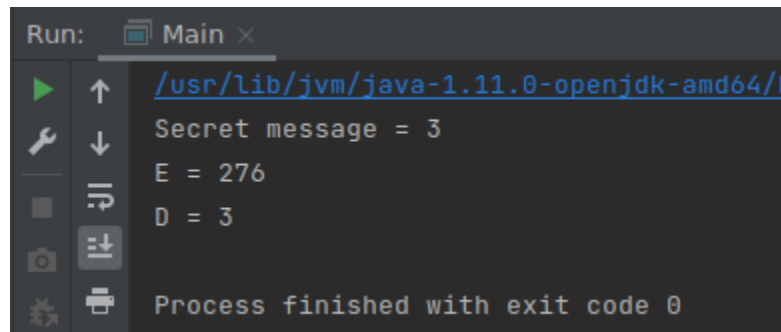
Скріншоти виконання:



```
Run: Main x  
/usr/lib/jvm/java-1.11.0-openjdk-amd64/b  
Secret message = 3  
E = 922  
D = 3  
Process finished with exit code 0
```



```
Run: Main x  
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java  
Secret message = 3  
E = 230  
D = 3  
Process finished with exit code 0
```



```
Run: Main x  
/usr/lib/jvm/java-1.11.0-openjdk-amd64/  
Secret message = 3  
E = 276  
D = 3  
Process finished with exit code 0
```

Висновки:

При виконанні даної лабораторної роботи проблем не виникло так як матеріал лекцій чудово покриває її тему.

Зважаючи на це, завдання лабораторної роботи виконано. Робота програми демонструє правильність обраних рішень.