**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЁТ**

**по лабораторной работе №3**

**«Асимметрическая криптография и цифровая подпись »**

*дисциплина: Математические основы защиты информации и информационной безопасности*

Вариант 3

Студент: Хиссен Али Уэддей

Группы: НПМмд-02-20

Ст/б: 10322009306

**МОСКВА**

**2021**

Оглавление

[Цель работы 3](#_Toc58687022)

[Описание алгоритма 3](#_Toc58687023)

[Реализация алгоритма (код программы) 6](#_Toc58687024)

[Результаты работы программы с различными исходными текстами, ключами 8](#_Toc58687025)

[Вывод 9](#_Toc58687026)

[Ответы на контрольные вопросы 9](#_Toc58687027)

# **Цель работы**

Целью работы является изучение алгоритмов шифрования с открытым ключом, а также реализация варианта асимметричного криптографического алгоритма Эль Гамаль.

Определила вариант с помощью формулы , где — номер студ.билета, N— количество заданий. Так как Sn=1032209306, а N=4, то номер варианта лабораторной работы 3.

# **Описание алгоритма**

Асимметрическая схема Эль Гамаль, предложенная автором (El Gamal), использует операцию возведения в степень по модулю простого числа. При этом трудноразрешимой задачей для злоумышленника является отыскание не числа, которое возведено в степень, а то, в какую степень возведено известное число. Эта задача носит название проблемы дискретного логарифма.

На этапе выработки ключей:

1. Выбирается произвольное (правда достаточно большое) простое число р.
2. Для этого простого числа определяется любой образующий элемент (англ. primitive root) – т. е. такое число а, при многократном возведении которого в степень по модулю р (, будут перебираться (в произвольном порядке, но обязательно по одному разу) все числа от 1 до (p-1) включительно.
3. Генерируется произвольное случайное число х (0 < х < p) – это и есть закрытый ключ.
4. Вычисляется значение – комбинация (a, p, b) представляет собой открытый ключ получателя.

На этапе шифрования:

1. Отправитель генерирует произвольное случайное число y (0< y <p).
2. Помещает в начале шифрограммы число ().
3. Вычисляет величину k = () = .
4. Используя некоторую, заранее оговоренную в данной реализации, часть k в качестве симметричного ключа для любого блочного шифра шифрует отправляемое сообщение.
5. Надежно стирает числа y и k из оперативной памяти и других мест, куда они могли случайно попасть.

На этапе дешифрования:

1. По приходу зашифрованного сообщения получатель отделяет от пакета величину () и вычисляет на ее основе – математика доказывает, что полученное число будет равно тому самому k, которое вычислил отправитель, так как в данной формуле операнды x и y можно менять местами.
2. Выделив из k туже самую часть, что и отправитель, получатель дешифрует весь идущий далее пакет симметричным алгоритмом. Схема алгоритма Эль Гамаль приведена ниже на рисунке 1.

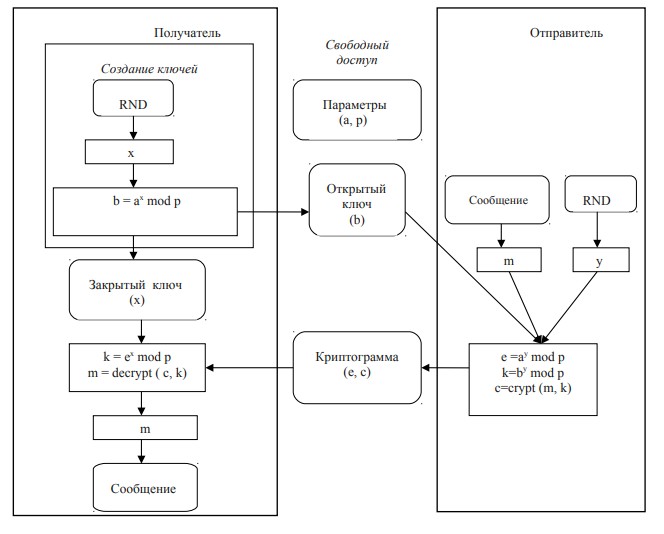


Рис.1 Криптосистема Эль Гамаль

Проблема дискретного логарифма состоит в том, что, зная основание степени и получившийся после возведения результат по модулю простого числа, невозможно за обозримое время определить, в какую именно степень было возведено основание. В схеме Эль Гамаль потенциальный злоумышленник может получить значения a, p, () и (). Однако из-за сложности определения чисел х и y "в чистом виде" у него не оказывается возможности вычислить значение k = (), которое так необходимо для прочтения шифровки.

По криптостойкости с схеме Эль Гамаль 512-битное число p приравнивается к 56-битному симметричному ключу, размер которого в настоящее время недостаточен для надежного шифрования. Поэтому на практике применяются р длиной в 768, 1024 и 1536 бит.

# **Реализация алгоритма (код программы)**

import java.math.BigInteger;

import java.security.SecureRandom; import java.util.Random;

public class Main {

static Random random = new SecureRandom(); static int p,g,y,x;

static String plaintext ="Proverka"; static int n=10;

public static void main(String[] args) { int[] text = plaintext.chars().toArray(); int[][] cipher = new int[text.length][2]; int[] decrypted = new int[text.length];

System.out.println("Ваш текст: "+plaintext); System.out.println("\nсоздаем ключи"); Random random = new SecureRandom();

p=BigInteger.probablePrime(n,random).intValue(); x=random.nextInt(p-2)+2;

g=pr\_root(p);

System.out.println("x="+x+" (закрытый ключ)"); y=pmod(g,x,p);

System.out.println("открытый ключ: p="+p+", g="+g+", y="+y);

for (int i=0;i<text.length;i++) { cipher[i]=encrypt(text[i]);

}

for (int i=0;i<cipher.length;i++) { decrypted[i]=decrypt(cipher[i]);

}

System.out.println("\nРасшифрованный текст:\n"); for (int i=0;i<decrypted.length;i++) { System.out.print((char)decrypted[i]);

}

}

static int[] encrypt(int T){

int k = random.nextInt(p-2)+2; int a = pmod(g,k,p);

int b = bi(y).pow(k).multiply(bi(T)).mod(bi(p)).intValue(); int block[] = {a,b};

return block;

}

static int decrypt(int[] block){ int a = block[0],b=block[1];

int T = inv(a,x,p).multiply(bi(b)).mod(bi(p)).intValue(); return T;

}

static BigInteger inv(int n,int pow,int m)

{

return (bi(n).pow(pow)).modInverse(bi(m));

}

static BigInteger bi(int x){ return BigInteger.valueOf(x);

}

static int pmod(int a,int b,int m){

return bi(a).modPow(bi(b), bi(m)).intValue();

}

static int pr\_root(int p){ int i;

for(i=2;i<p;i++){ if(pmod(i,p-1,p)==1){ boolean flag = true; for(int j=1;j<p-1;j++){ if(pmod(i,j,p)==1){ flag=false; break;

}

}

if(flag){ break;

}

}

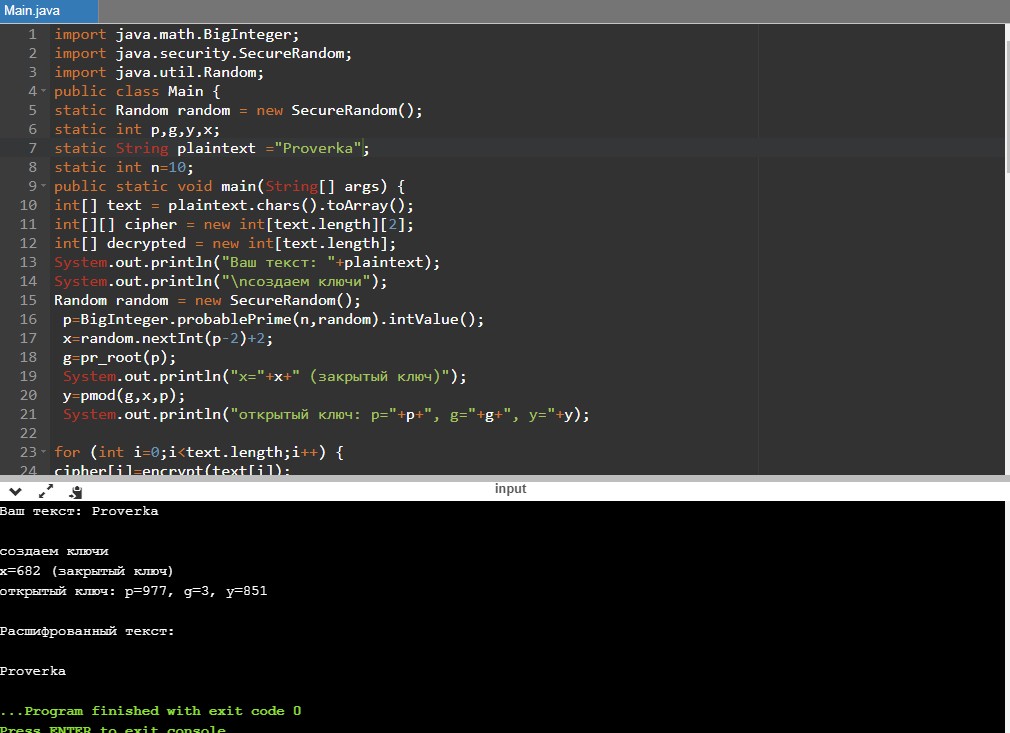
}

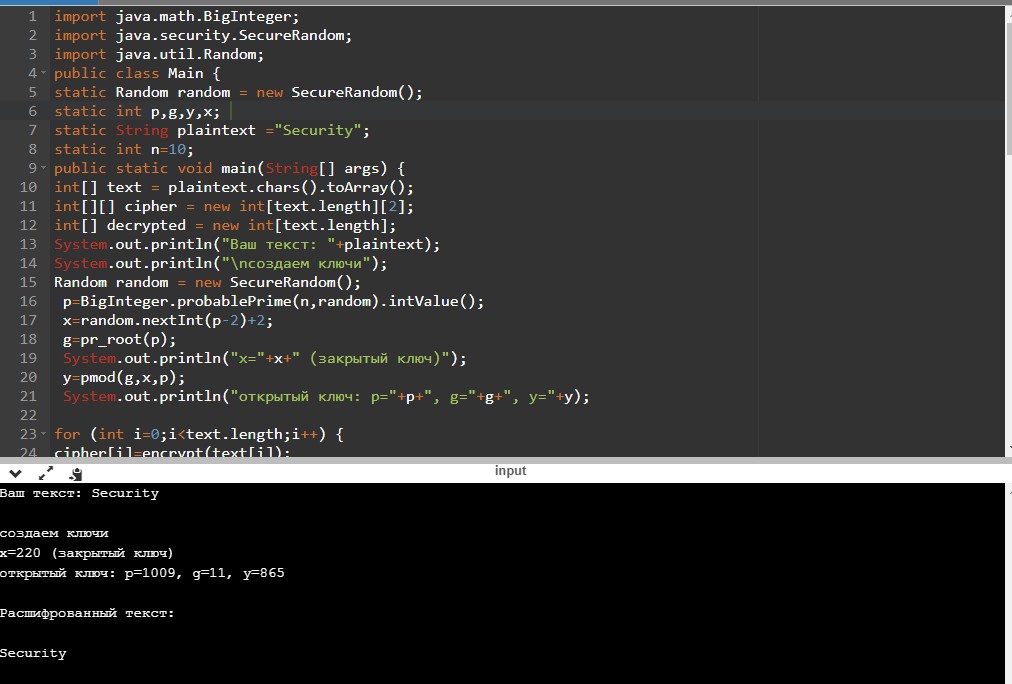
return i;

}

}

# **Результаты работы программы с различными исходными текстами, ключами**





# **Вывод**

В ходе выполнения лабораторной работы были изучены алгоритмы шифрования с открытым ключом. Также реализован вариант асимметричного криптографического алгоритма Эль Гамаль.

# **Ответы на контрольные вопросы**

1. **Какая процедура является более производительной – асимметричное шифрование/ дешифрование или симметричное шифрование/дешифрование?**

Симметричное шифрование/дешифрование является более производительной процедурой, т.к все алгоритмы асимметричного шифрования используют медленные операции возведения в степень по модулю, нахождение обратных элементов

1. **К какому типу криптоалгоритма (с точки зрения его устойчивости к взлому) и почему относится алгоритм RSA?**

Алгоритм RSA относится к вычислительно стойким алгоритмам шифрования, так в основе его лежит задача факторизации большого числа, решение которой перебором (а других эффективных алгоритмов не существует) занимает нецелесообразно длительное время.

1. **Какая трудноразрешимая математическая задача лежит в основе стойкости алгоритма RSA?**

В основе алгоритма лежит трудоемкая с точки зрения времени работы задача разложения числа как произведение двух больших простых чисел.

1. **Какая трудноразрешимая математическая задача лежит в основе стойкости алгоритма Эль Гамаль?**

В основе алгоритма Эль Гамаль лежит трудноразрешимая задача поиска степени, в которое возведено известное число. Эта задача носит название проблемы дискретного логарифма.

1. **В чем заключается проблема дискретного логарифма?**

Проблема дискретного логарифма состоит в том, что, зная основание степени и получившийся после возведения результат по модулю простого числа, невозможно за обозримое время определить, в какую именно степень было возведено основание. В схеме Эль Гамаль потенциальный злоумышленник может получить значения и . Однако из-за сложности определения чисел х и y "в чистом виде" у него не оказывается возможности вычислить значение , которое так необходимо для прочтения шифровки.

1. **В чем заключаются проблемы разложения больших чисел на простые множители и вычисления корней алгебраических уравнений?**

Проблема заключается в отсутствии эффективных алгоритмов разложения числа на простые множители. Проблема вычисления корней некоторых (специальных) алгебраических уравнений также заключается в отсутствии эффективных алгоритмов их решения без обладания дополнительной информацией, делающей их решение несложной и быстро решаемой задачей.