МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Алгоритм Хьюза (Hudges)**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Таранова Алексея Вадимовича

|  |  |  |
| --- | --- | --- |
| Преподаватель  аспирант | \_\_\_\_\_\_\_\_\_\_ | Р. А. Фарахутдинов |
|  | подпись, дата |  |

Саратов 2023

**1. Теоретическая часть**

Цель работы – изучение алгоритма Хьюза и его программная реализация.

В криптографической практике при каждом сеансе связи принято пользоваться новым ключом шифрования. Этот ключ называют сеансовым. Такое ограничение на существование ключа является важным условием надёжности конфиденциальной связи, но в то же время создаёт дополнительную сложную задачу передачи сеансового ключа.

Чтобы уменьшить утечку информации о ключах и облегчить процесс генерации ключей создают протоколы распределения ключей только по запросу на очередной сеанс связи.

В двухпроходном протоколе Диффи и Хеллмана безопасность протокола основывается на трудоемкости вычисления дискретных логарифмов в конечном поле, в сравнении с легкостью возведения в степень в том же самом поле. Он предполагает предварительный выбор некоторых начальных параметров, которые должны быть доступны всем участникам протокола и сети. Такие параметры будем называть общими параметрами.

В протоколе Диффи-Хеллмана, по сути, секретный ключ генерируется по ходу его проведения, и никто не знает на сколько хорошими характеристиками он будет обладать, что является одним из заметных недостатков этого протокола.

Иной вариант алгоритма Диффи-Хеллмана, предложенный Хьюзом (Hughes), позволяет Алисе сначала генерировать ключ проверить его надёжность, и уже потом послать его Бобу. Общие параметры те же. Он так же двухпроходный.

Алгоритм Хьюза:

1. где – случайное секретное целое число Алисы из интервала (Алиса генерирует сеансовый ключ K).

, где – случайное секретное целое число Боба из интервала с условием . Если сильное простое число, например вида , тогда может быть любым большим случайным нечетным числом, кроме .

3.

4. .

Если все выполнено правильно, то . Действительно, учитывая свойства индексов:

Преимуществом описанного выше протокола над протоколом Диффи -Хеллмана состоит в том, что K можно вычислить заранее, до какого-либо взаимодействия, и Алиса может зашифровать сообщения с помощью K задолго до установления соединения с Бобом. При этом Алиса может послать сообщение сразу множеству людей, а передать ключ позднее каждому по отдельности.

**2. Описание программы**

Код включает три роли: Алиса, Боб и центр генерации, который устанавливает начальные параметры (роли описаны в public static void main(String[] args))

Центр генерации (роль 2) генерирует сильное простое число P (используя private static BigInteger generateStrongPrime()) и генератор G в качестве параметров алгоритма Хьюза. Параметры P, G и q (случайно сгенерированное простое число) сохраняются в текстовых файлах (для работы с файлами используются writeFile и readFile).

Сначала в роли Алисы (роль 0) вычисляется сеансовый ключ (выбор 0). Затем Боб (роль 1) высчитывает и отправляет открытый ключ Алисе (выбор 1). Алиса (роль 0) отправляет свой открытый ключ X Бобу (выбор 1), после чего Боб (роль 1) вычисляет сеансовый ключ (выбор 0).

Если все выполнено правильно, то сеансовые ключи совпадут.

**3. Пример запуска программы**

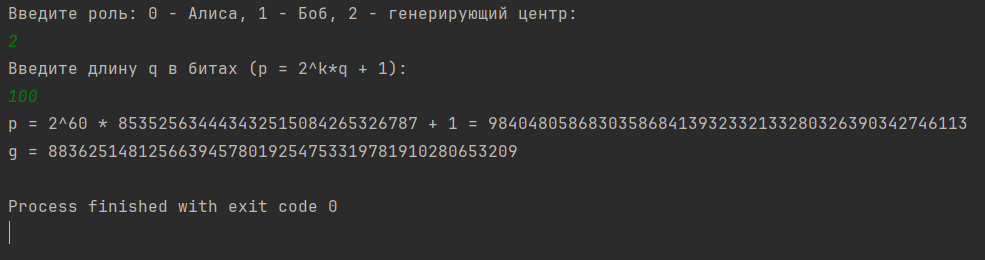


Рисунок 1 – Генерация p, g, q

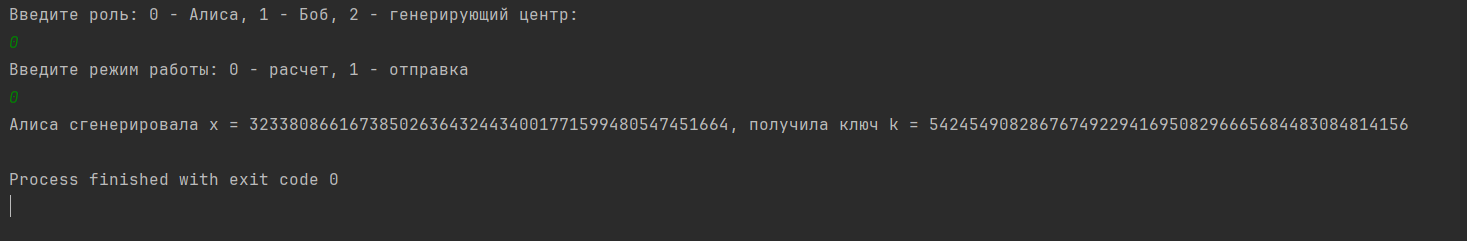


Рисунок 2 – Алиса cгенерировала сеансовый ключ K

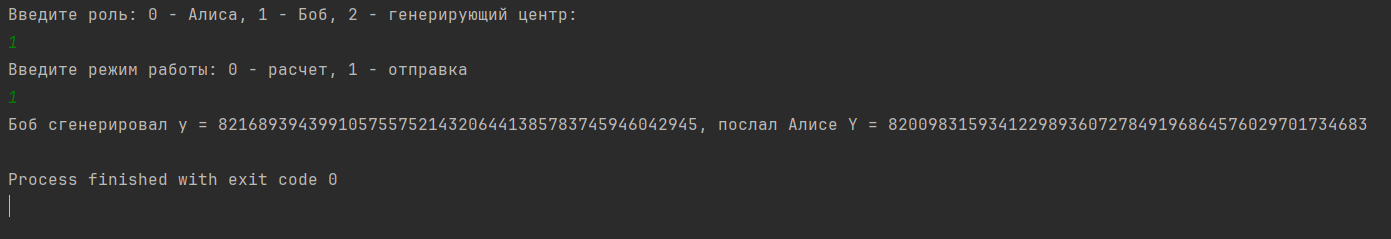


Рисунок 3 – Боб сгенерировал секретное число y и отправил Алисе Y

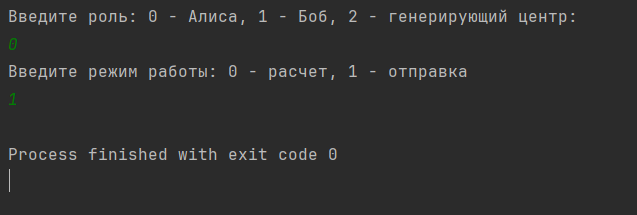


Рисунок 4 – Алиса отправила Бобу X.

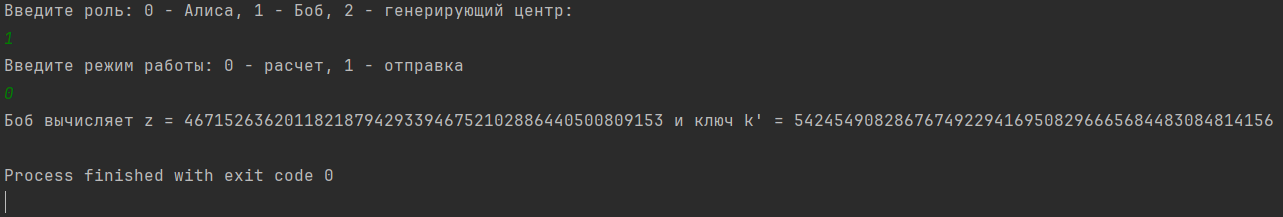


Рисунок 5 – Боб вычисляет z и ключ K’. Ключи K и K’ совпали

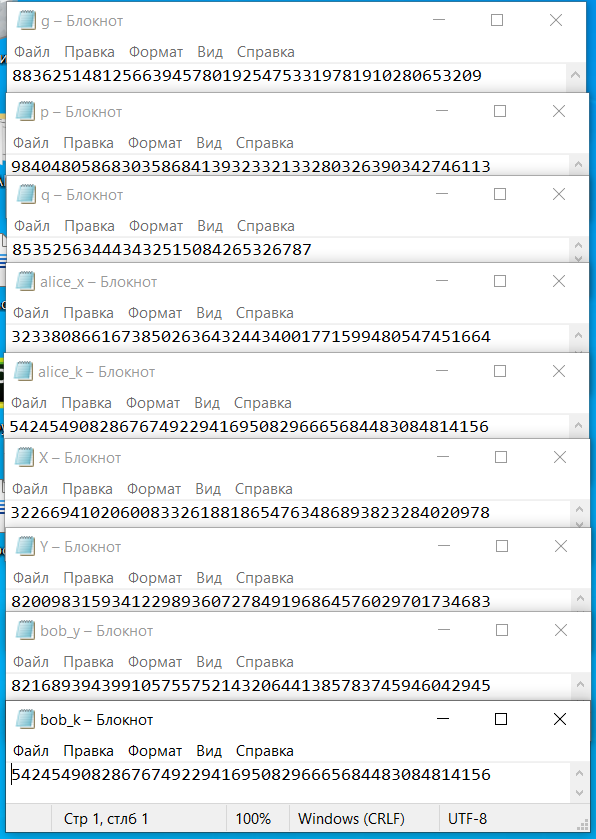


Рисунок 6 – Содержимое файлов, использовавшихся для хранения переменных

**ПРИЛОЖЕНИЕ А**

Листинг 1 – Hughes.java

package com.aleksey;

import java.awt.\*;

import java.io.\*;

import java.math.BigInteger;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.Scanner;

public class Hughes {

private static final class Alice {

BigInteger x;

BigInteger Y;

BigInteger k;

}

private static final class Bob {

BigInteger y;

BigInteger X;

BigInteger k;

BigInteger z;

}

private static BigInteger generateStrongPrime() { // Генерация сильного простого числа p

q = BigInteger.probablePrime(qSize, new Random()); // q вероятно-случайное (тест Миллера-Рабина), размером qSize бит

BigInteger mod = BigInteger.probablePrime(2048, new Random()); // степень 2 вероятно-случайная

BigInteger k = BigInteger.TWO;

BigInteger res;

while (true) {

res = BigInteger.TWO.modPow(k, mod).multiply(q).add(BigInteger.ONE); //вычисление p=(2^k)\*q+1

if (res.isProbablePrime(20)) { //проверка на простое

System.out.print("p = 2^" + k + " \* " + q + " + 1 = ");

return res; //вывод

}

k = k.add(BigInteger.ONE); // k + 1

}

}

private static BigInteger q;

private static int qSize = 32;

public static BigInteger P;

public static BigInteger G;

private static void writeFile (String text, String filename) {

try {

File output = new File(filename);

Files.deleteIfExists(Paths.get(filename));

if (output.createNewFile()) {

BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(output), "UTF-8"));

out.write(text);

out.close();

} else {

System.out.println("Файл с таким именем уже существует или нет прав доступа к директории.");

System.exit(-1);

}

} catch (IOException e ) {

e.printStackTrace();

}

}

private static List<BigInteger> readFile (String filename) {

List<BigInteger> res = new ArrayList<>();

try {

File file = new File(filename);

FileReader fr = new FileReader(file);

BufferedReader reader = new BufferedReader(fr);

String line = reader.readLine();

while (line != null) {

res.add(new BigInteger(line));

line = reader.readLine();

}

fr.close();

} catch (IOException e) {

System.out.println("Не найден файл " + filename);

System.exit(-1);

}

return res;

}

public static void main(String[] args) {

String aliceFolder = args[0]; // Папки, где находятся файлы с g, p, q, X, Y, и т. д.

String bobFolder = args[1];

String sharedFolder = args[2];

Scanner in = new Scanner(System.in);

System.out.println("Введите роль: 0 - Алиса, 1 - Боб, 2 - генерирующий центр:");

int role = in.nextInt();

switch (role) {

case 0 -> { // Часть Алисы

Alice alice = new Alice();

System.out.println("Введите режим работы: 0 - расчет, 1 - отправка");

int mode = in.nextInt();

P = readFile(sharedFolder.concat("\\p")).get(0);

G = readFile(sharedFolder.concat("\\g")).get(0);

if (mode == 0) { // 1 пункт алгоритма, Алиса генерирует сеансовый ключ (расчет)

String alice\_k\_file = "\\alice\_k";

alice.x = new BigInteger(2048, new Random()).mod(P); // x из интервала от 1 до p

if (alice.x.compareTo(BigInteger.TWO) <= 0)

alice.x.add(BigInteger.ONE);

alice.k = G.modPow(alice.x, P); // alice\_k = g^x mod p

System.out.println("Алиса сгенерировала x = " + alice.x + ", получила ключ k = " + alice.k);

writeFile(alice.k.toString(), aliceFolder.concat(alice\_k\_file));

writeFile(alice.x.toString(), aliceFolder.concat("\\alice\_x"));

} else { // 3 пункт алгоритма, A -> B

String xFile = "\\X";

BigInteger Y = readFile(bobFolder.concat("\\Y")).get(0);

BigInteger x = readFile(bobFolder.concat("\\alice\_x")).get(0);

BigInteger X = Y.modPow(x, P);

writeFile(X.toString(), bobFolder.concat(xFile));

System.out.println("X = " + X);

}

}

case 1 -> { // Часть Боба

Bob bob = new Bob();

System.out.println("Введите режим работы: 0 - расчет, 1 - отправка");

int mode = in.nextInt();

P = readFile(sharedFolder.concat("\\p")).get(0);

G = readFile(sharedFolder.concat("\\g")).get(0);

if (mode == 0) { // 4 пункт алгоритма, сеансовый ключ Боба K'

bob.y = readFile(bobFolder.concat("\\bob\_y")).get(0);

bob.X = readFile(bobFolder.concat("\\X")).get(0);

bob.z = bob.y.modInverse(P.subtract(BigInteger.ONE));

bob.k = bob.X.modPow(bob.z, P);

writeFile(bob.k.toString(), aliceFolder.concat("\\bob\_k"));

System.out.println("Боб вычисляет z = " + bob.z + " и ключ k' = " + bob.k);

} else { // 2 пункт алгоритма, B -> A

bob.y = BigInteger.probablePrime(256, new Random()).mod(P);

BigInteger Y = G.modPow(bob.y, P);

System.out.println("Боб сгенерировал y = " + bob.y + ", послал Алисе Y = " + Y);

writeFile(Y.toString(), aliceFolder.concat("\\Y"));

writeFile(bob.y.toString(), bobFolder.concat("\\bob\_y"));

}

}

case 2 -> { // Часть генерирующего центра

System.out.println("Введите длину q в битах (p = 2^k\*q + 1):");

qSize = in.nextInt();

P = generateStrongPrime(); // Генерация сильного простого числа p

System.out.println(P);

Random r = new Random();

G = BigInteger.probablePrime(P.bitLength() - 4, r);

System.out.println("g = " + G);

writeFile(P.toString(), sharedFolder.concat("\\p"));

writeFile(G.toString(), sharedFolder.concat("\\g"));

writeFile(q.toString(), sharedFolder.concat("\\q"));

}

}

}

}