Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

09.03.01 "Информатика и вычислительная техника" профиль "Программное обеспечение средств

вычислительной техники и автоматизированных систем"

**ОТЧЕТ**

Расчётно-графической работе

Вариант 3

Протокол Фиата-Шамира

Выполнил:

студент гр.ИВ-922 \_\_\_\_\_\_\_\_\_\_\_\_\_/Мустафин Р. И./

«10» октября 2022г.

Принял:

Ассистент кафедры ПМиК \_\_\_\_\_\_\_\_\_\_\_\_\_/Меркулов И. А./

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новосибирск, 2022

Оглавление

[Задание 3](#_Toc116295852)

[Описание работы 4](#_Toc116295853)

[Листинг 5](#_Toc116295854)

# Задание

Для выполнения этого варианта задания необходимо разработать клиент-серверное приложение с авторизацией по протоколу Фиата-Шамира. Открытые ключи с соответствующими логинами должны храниться в файле (или базе данных) на сервере, клиентское приложение при этом не должно отправлять на сервер никаких закрытых данных, закрытый ключ нигде не хранится и используется исключительно для осуществления работы протокола с клиентской стороны. Все открытые параметры системы рассылаются сервером при установке соединения с клиентом.

# Описание работы

Программа была разделена на клиент и сервер. В сервере генерируются неравные простые большие числа P и Q, и число N = P \* Q

Клиент генерирует пароль S взаимнопростой с N. Затем вычисляет V = S^2%N. Это открытый ключ. Клиент хочет доказать, что она знает S из которого получился V. Для этого мы t раз выполняем следующие действия:

1. Генерируем число r в диапазоне от 1 до N-1, вычисляем x=r^2%N и передаём его серверу
2. Сервер генерирует число е в диапазоне от 0 до 1 и отправляет клиенту
3. Клиент вычисляет y=r\*s^e и отправляет серверу.

Сервер проверяет равенство у^2 = x\*v^e (%N). Получаем результат и повторяем эти шаги t раз

Изображение выглядит как текст

Автоматически созданное описание

# Листинг

using System;

using System.Numerics;

namespace lab1

{

class RGRClient

{

private Random \_random = new Random();

private RGRServer \_server = new RGRServer();

private BigInteger \_s;

private BigInteger \_v;

private BigInteger \_t;

public RGRClient()

{

\_s = \_random.Next((int)Math.Sqrt((double)\_server.N), (int)Math.Sqrt((double)\_server.N) + (int)Math.Pow(10, 4));

\_v = BigInteger.Pow(\_s, 2) % \_server.N;

\_t = 1000;

}

public bool Login()

{

for (int i = 0; i < \_t; i++)

{

BigInteger r = \_random.Next(1, (int)\_server.N - 1);

BigInteger x = BigInteger.Pow(r, 2) % \_server.N;

BigInteger e = \_server.GenerateE(x);

Console.WriteLine(r);

BigInteger y = r \* BigInteger.Pow(\_s, (int)e);

if (\_server.Verify(y, \_v) == false)

{

return false;

}

}

return true;

}

}

class RGRServer

{

private Random \_random = new Random();

private CriptoHelper \_criptoHelper = new CriptoHelper();

private Criptographic \_criptographic = new Criptographic();

private BigInteger \_p;

private BigInteger \_q;

private BigInteger \_n;

private BigInteger \_e;

private BigInteger \_clientX;

public BigInteger N => \_n;

public RGRServer()

{

\_p = \_criptoHelper.GetPrimeRandomNumber(257, 3);

\_q = \_criptoHelper.GetPrimeRandomNumber(257, 3);

\_n = \_p \* \_q;

}

public BigInteger GenerateE(BigInteger x)

{

\_clientX = x;

return \_e = \_random.Next(0, 2);

}

public bool Verify(BigInteger y, BigInteger v)

{

if (BigInteger.Pow(y, 2) % N != \_clientX \* BigInteger.Pow(v, (int)\_e) % N)

{

return false;

}

return true;

}

}

}