Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

РГР «Доказательство с нулевым знанием»  
по дисциплине «Защита информации»

Вариант №11(1)

Выполнил:

студент группы ИП-811

Миронеко К.А.

Работу проверил:

Доцент кафедры ПМиК  
Ракитский А.А.

Оглавление

[Задание 3](#_Toc91330529)

[Результаты 4](#_Toc91330530)

[Листинг 7](#_Toc91330531)

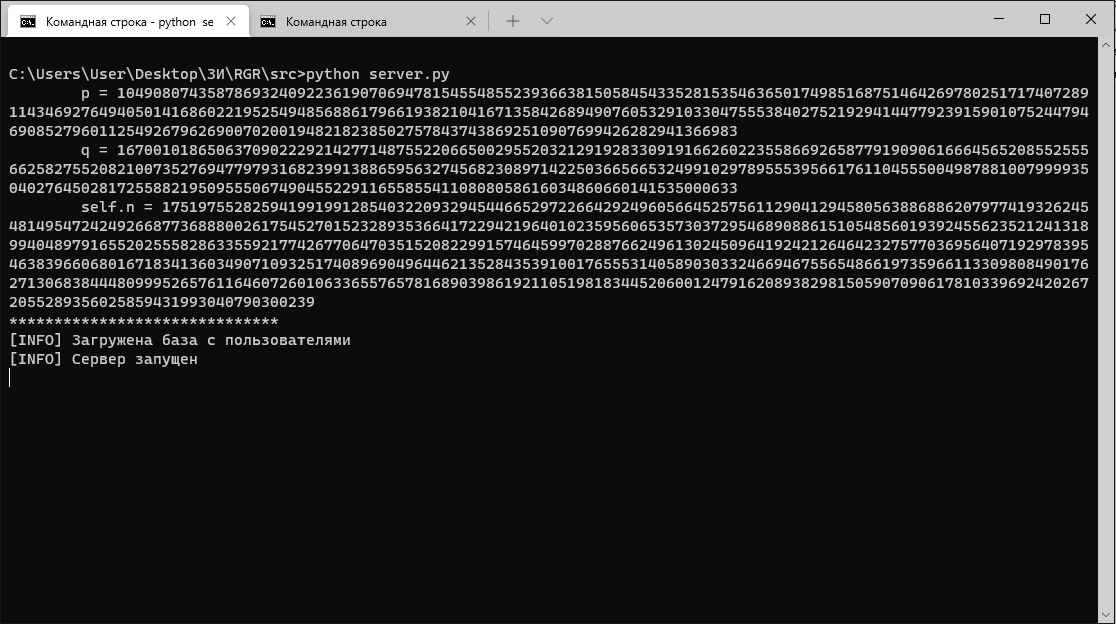
# Задание

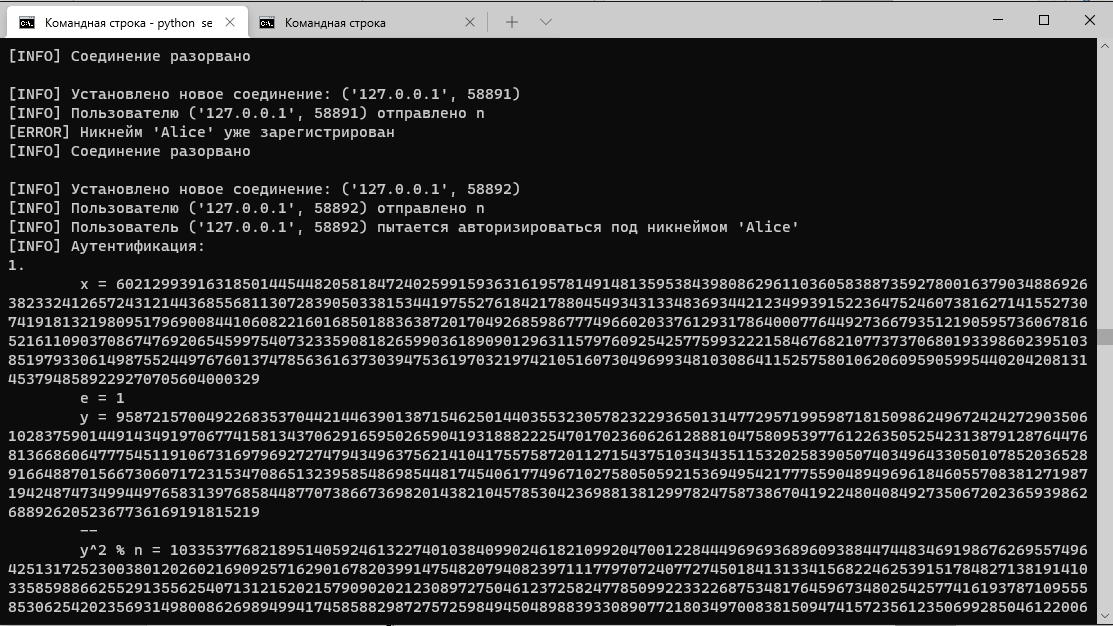
Необходимо написать программу, реализующую протокол доказательства с нулевым знанием Фиата-Шамира.

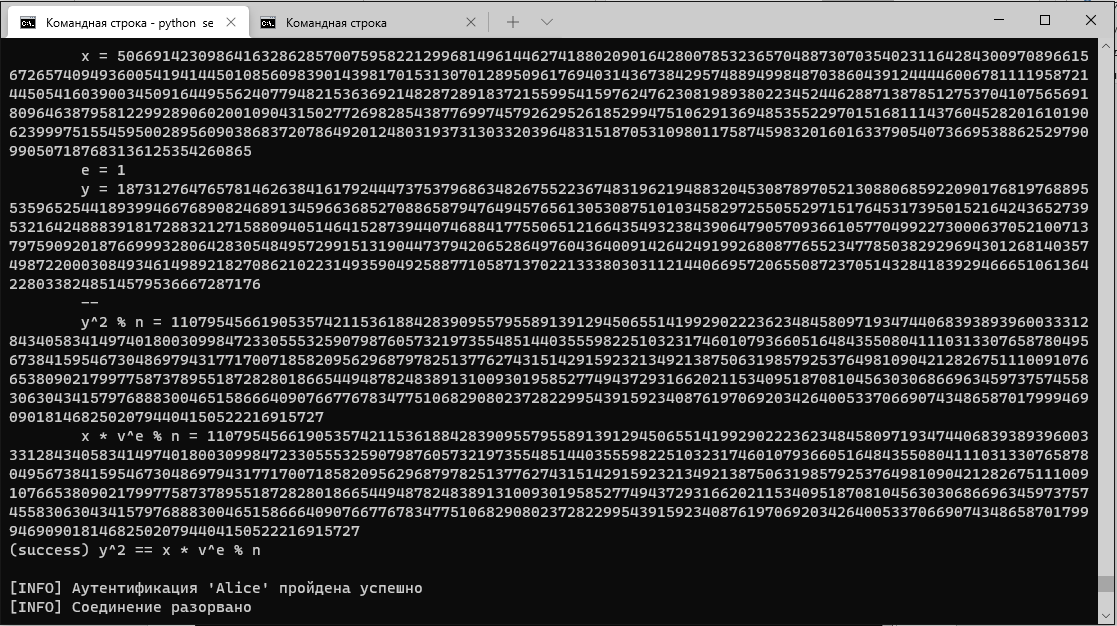
Для выполнения этого варианта задания необходимо разработать клиент-серверное приложение с авторизацией по протоколу Фиата-Шамира.

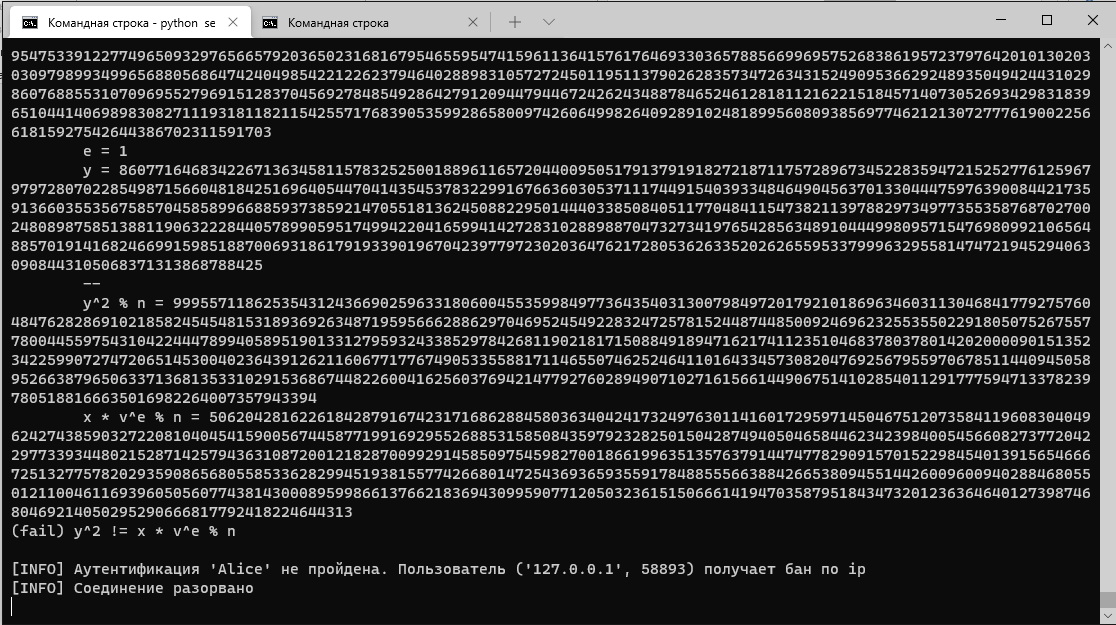
Открытые ключи с соответствующими логинами должны храниться в файле (или базе данных) на сервере, клиентское приложение при этом не должно отправлять на сервер никаких закрытых данных, закрытый ключ нигде не хранится и используется исключительно для осуществления работы протокола с клиентской стороны. Все открытые параметры системы рассылаются сервером при установке соединения с клиентом.

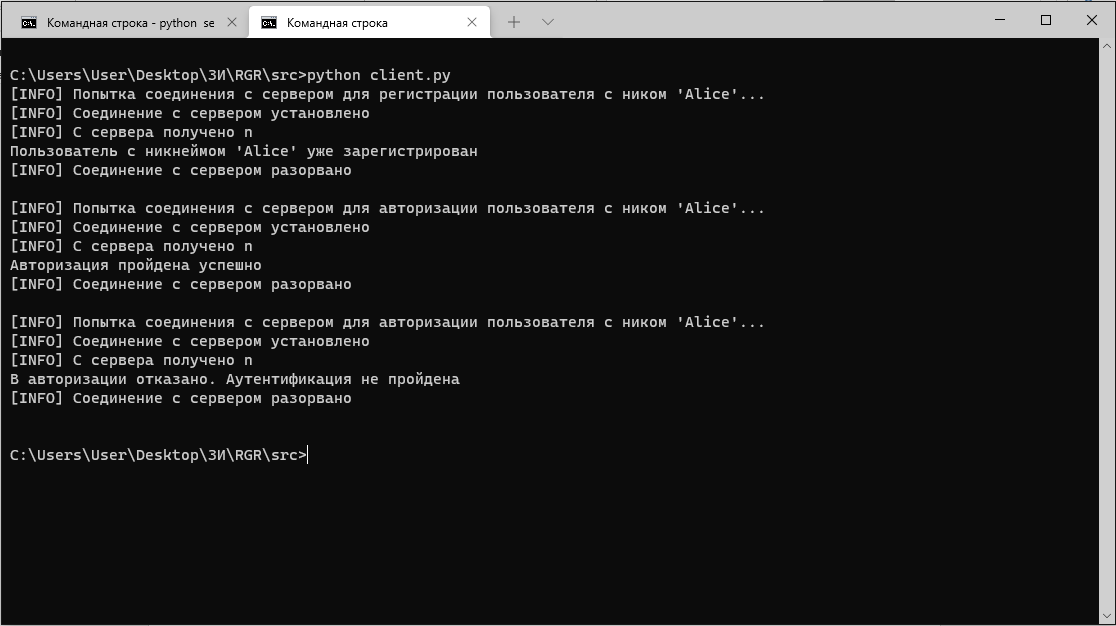
# Результаты











# Листинг

**server.py**

import json

from lib import \*

import socket

import signal

class Server:

def \_\_init\_\_(self, host: str = "localhost", port: int = 3000):

self.\_host = host

self.\_port = port

self.\_buffer\_size = 1024

p = q = gen\_prime(1 << 1023, (1 << 1024) - 1)

while p == q:

q = gen\_prime(1 << 1023, (1 << 1024) - 1)

self.n = p \* q

print(f"\t{p = }", f"\t{q = }", f"\t{self.n = }", '\*' \* 30, sep='\n')

try:

with open("registered\_users.json", "r") as f:

self.\_registered\_users = json.load(f)

print("[INFO] Загружена база с пользователями")

except FileNotFoundError:

with open("registered\_users.json", "w") as f:

self.\_registered\_users = dict()

json.dump(self.\_registered\_users, f)

print(

"[ERROR] Не удалось загрузить базу с пользователями. Была создана новая")

def run(self):

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.bind((self.\_host, self.\_port))

sock.listen()

print("[INFO] Сервер запущен")

while True:

conn, addr = sock.accept()

print(f"\n[INFO] Установлено новое соединение: {addr}")

conn.send((bytes(str(self.n), encoding = "utf8")))

print(f"[INFO] Пользователю {addr} отправлено n")

data = conn.recv(self.\_buffer\_size).decode("utf8")

if data == "{register}":

name = conn.recv(self.\_buffer\_size).decode("utf8")

v = int(conn.recv(self.\_buffer\_size).decode("utf8"))

if name not in self.\_registered\_users.keys():

print(f"[INFO] Пользователь {addr} зарегистрировался с никнеймом '{name}' и ключом {v}")

self.\_registered\_users[name] = v

with open("registered\_users.json", "w") as f:

json.dump(self.\_registered\_users, f)

conn.send((bytes("success", encoding = "utf8")))

else:

print(f"[ERROR] Никнейм '{name}' уже зарегистрирован")

conn.send((bytes("already registered", encoding = "utf8")))

if data == "{auth}":

name = conn.recv(self.\_buffer\_size).decode("utf8")

v = self.\_registered\_users[name]

print(f"[INFO] Пользователь {addr} пытается авторизироваться под никнеймом '{name}'")

print(f"[INFO] Аутентификация:")

t = 20

for i, \_ in enumerate(range(t), 1):

x = int(conn.recv(self.\_buffer\_size).decode("utf8"))

e = random.randint(0, 1)

conn.send((bytes(str(e), encoding = "utf8")))

y = int(conn.recv(self.\_buffer\_size).decode("utf8"))

y2 = exponentiation\_modulo(y, 2, self.n)

xv = x \* exponentiation\_modulo(v, e, self.n) % self.n

print(f"{i}.\n\t{x = }\n\t{e = }\n\t{y = }\n\t--\n\ty^2 % n = {y2}\n\tx \* v^e % n = {xv}")

if y2 == xv:

print(f"(success) y^2 == x \* v^e % n\n")

if i == t:

print(f"[INFO] Аутентификация '{name}' пройдена успешно")

conn.send((bytes("success", encoding = "utf8")))

else:

conn.send((bytes("check", encoding = "utf8")))

else:

print(f"(fail) y^2 != x \* v^e % n\n")

conn.send((bytes("fail", encoding = "utf8")))

print(f"[INFO] Аутентификация '{name}' не пройдена. Пользователь {addr} получает бан по ip")

break

conn.close()

print(f"[INFO] Соединение разорвано")

if \_\_name\_\_ == "\_\_main\_\_":

signal.signal(signal.SIGINT, signal.SIG\_DFL)

server = Server()

server.run()

**client.py**

import random

from lib import \*

import socket

random.seed(1)

class Client:

def \_\_init\_\_(self, name: str):

self.name = name

self.\_buffer\_size = 1024

def register(self, host: str = "localhost", port: int = 3000):

print(f"[INFO] Попытка соединения с сервером для регистрации пользователя с ником '{self.name}'...")

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.connect((host, port))

print(f"[INFO] Соединение с сервером установлено")

n = int(sock.recv(self.\_buffer\_size).decode("utf8"))

print(f"[INFO] С сервера получено n")

sock.send((bytes("{register}", encoding = "utf8")))

sock.send((bytes(self.name, encoding = "utf8")))

self.s = gen\_mutually\_prime\_big(n)

v = exponentiation\_modulo(self.s, 2, n)

sock.send((bytes(str(v), encoding = "utf8")))

status = sock.recv(self.\_buffer\_size).decode("utf8")

if status == "success":

print(f"Вы были успешно зарегистрированы")

elif status == "already registered":

print(f"Пользователь с никнеймом '{self.name}' уже зарегистрирован")

print(f"[INFO] Соединение с сервером разорвано\n")

sock.close()

def auth(self, host: str = "localhost", port: int = 3000):

print(f"[INFO] Попытка соединения с сервером для авторизации пользователя с ником '{self.name}'...")

sock = socket.socket()

sock.connect((host, port))

print(f"[INFO] Соединение с сервером установлено")

sock.send((bytes("{auth}", encoding = "utf8")))

n = int(sock.recv(self.\_buffer\_size).decode("utf8"))

print(f"[INFO] С сервера получено n")

sock.send((bytes(self.name, encoding = "utf8")))

while True:

r = random.randrange(1, n - 1)

x = exponentiation\_modulo(r, 2, n)

sock.send((bytes(str(x), encoding = "utf8")))

e = int(sock.recv(self.\_buffer\_size).decode("utf8"))

y = r \* self.s \*\* e % n

sock.send((bytes(str(y), encoding = "utf8")))

status = sock.recv(self.\_buffer\_size).decode("utf8")

if status == "success":

print("Авторизация пройдена успешно")

break

elif status == "fail":

print("В авторизации отказано. Аутентификация не пройдена")

break

elif status == "check":

pass

print(f"[INFO] Соединение с сервером разорвано\n")

sock.close()

if \_\_name\_\_ == '\_\_main\_\_':

alice = Client('Alice')

alice.register()

alice.auth()

# Мошенник, пытающийся авторизоваться как Alice

cheater = Client('Alice')

cheater.s = 1000

cheater.auth()

**lib.py**

import math

import random

import sys

def exponentiation\_modulo(a: int, x: int, p: int) -> int:

if p == 0:

raise ValueError("Модуль не может быть равен нулю")

if x < 0:

raise ValueError("Показатель не может быть отрицательным")

result = 1

a = a % p

if a == 0:

return 0

while x > 0:

if x & 1 == 1:

result = (result \* a) % p

a = (a \*\* 2) % p

x >>= 1

return result

def is\_prime(p: int, trials: int=20) -> bool:

if p == 2 or p == 3:

return True

if p < 2 or not (p & 1):

return False

for \_ in range(trials):

a = random.randint(2, p - 1)

if exponentiation\_modulo(a, (p - 1), p) != 1 or math.gcd(p, a) > 1:

return False

return True

def gen\_prime(left: int, right: int) -> int:

while True:

p = random.randint(left, right)

if is\_prime(p):

return p

def gen\_safe\_prime(a: int, b: int) -> int:

if a > b:

a, b = b, a

while True:

q = gen\_prime(a // 2, (b - 1) // 2)

if is\_prime(p := q \* 2 + 1):

return p

def gen\_g(p: int) -> int:

while True:

g = random.randrange(2, p)

if pow(g, (p - 1) // 2, p) != 1:

return g

def gen\_mutually\_prime(p: int):

while True:

if math.gcd(p, b := random.randrange(2, p)) == 1:

return b

def gen\_mutually\_prime\_big(p: int):

while True:

if math.gcd(p, b := random.randrange(p// 2, p)) == 1:

return b

def generalized\_euclidean\_algorithm(a: int, b: int) -> list[int, int, int]:

if a <= 0 or b <= 0:

raise ValueError("Числа могут быть только натуральными")

if a > b:

a, b = b, a

u = [a, 1, 0]

v = [b, 0, 1]

while v[0] != 0:

q = u[0] // v[0]

t = [u[0] % v[0], u[1] - q \* v[1], u[2] - q \* v[2]]

u, v = v, t

return u

def inverse(n, p):

gcd, inv, \_ = generalized\_euclidean\_algorithm(n, p)

assert gcd == 1

if inv < 0 :

inv += p

return inv