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.

Новосибирск 2021

Оглавление

Задание 3
Результаты 4
Листинг 7

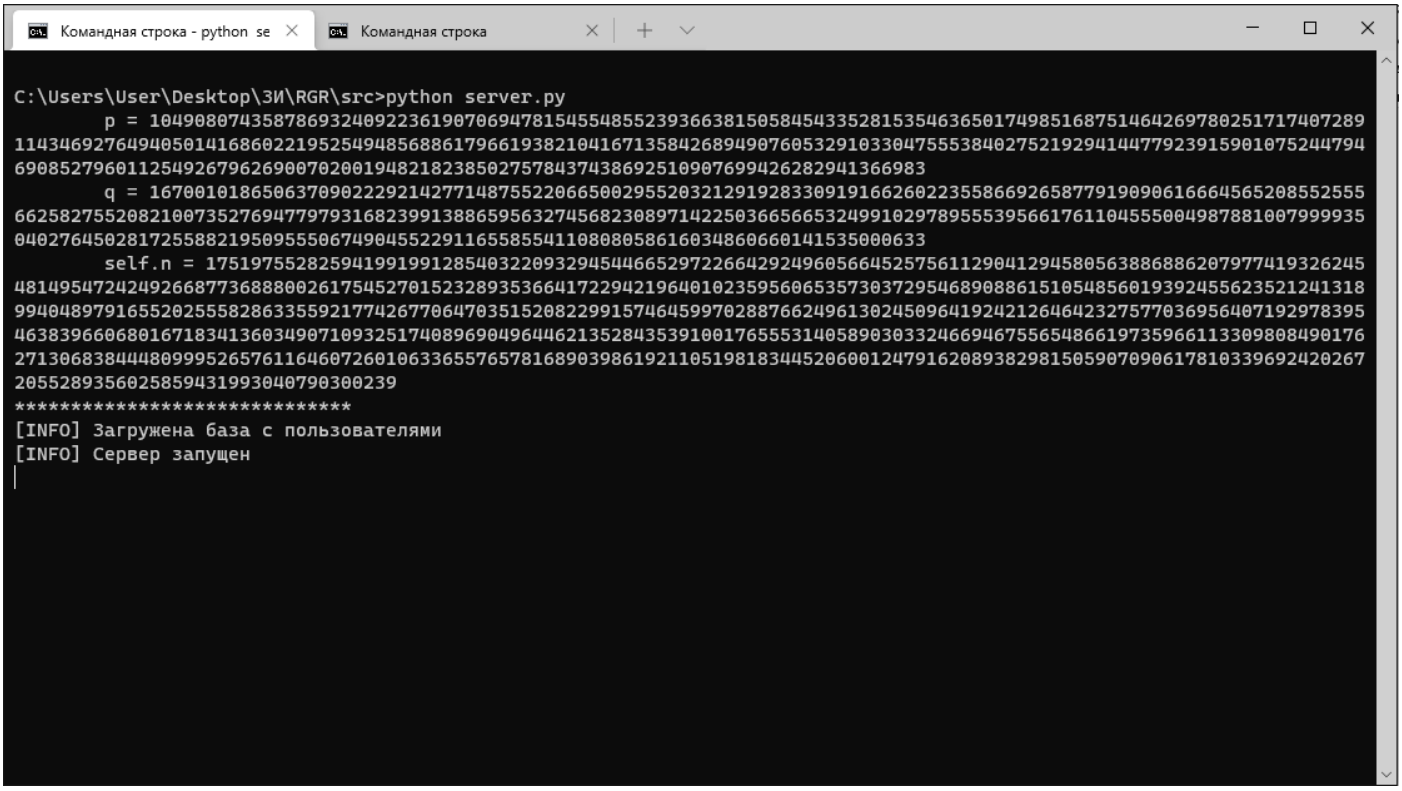
Задание

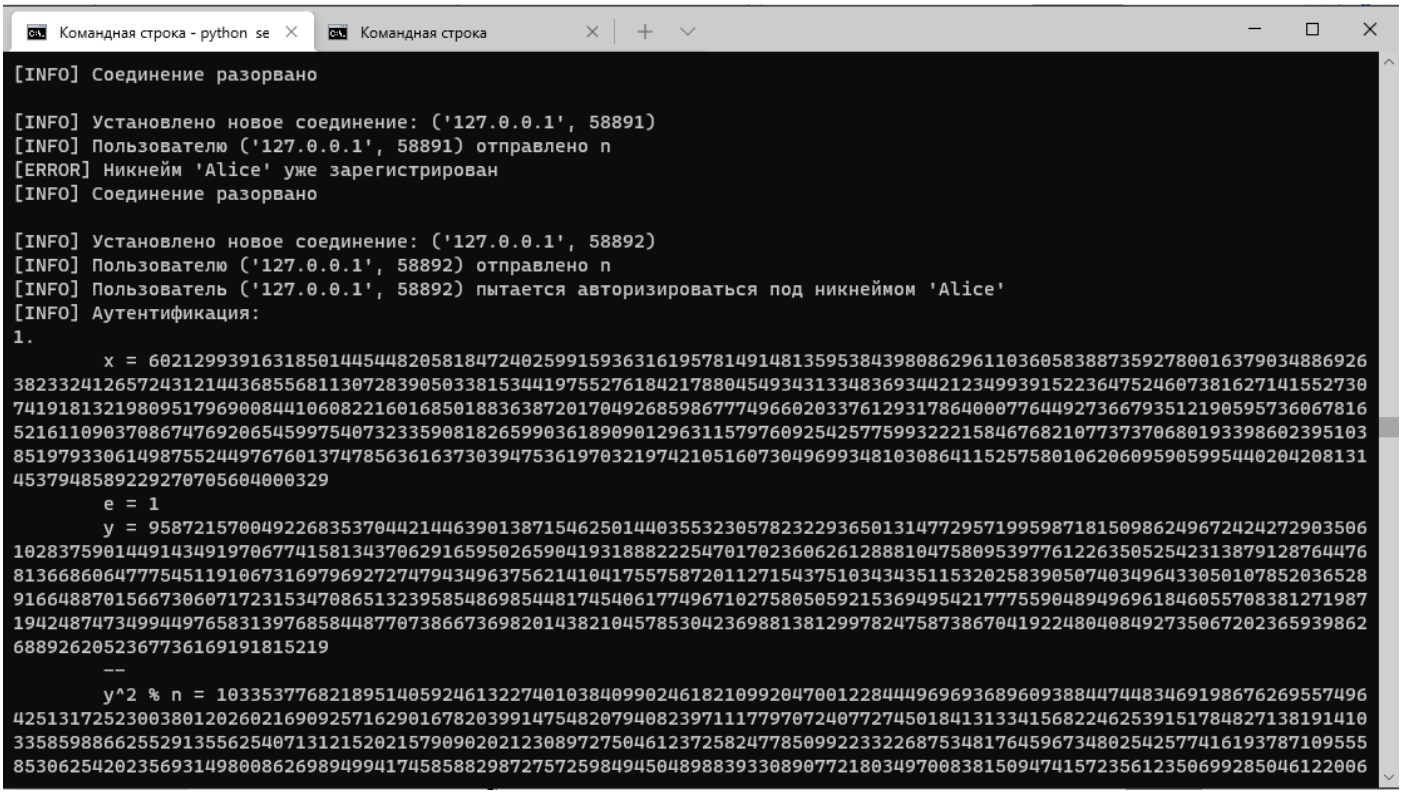
Необходимо написать программу, реализующую протокол доказательства с нулевым знанием Фиата-Шамира.

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.

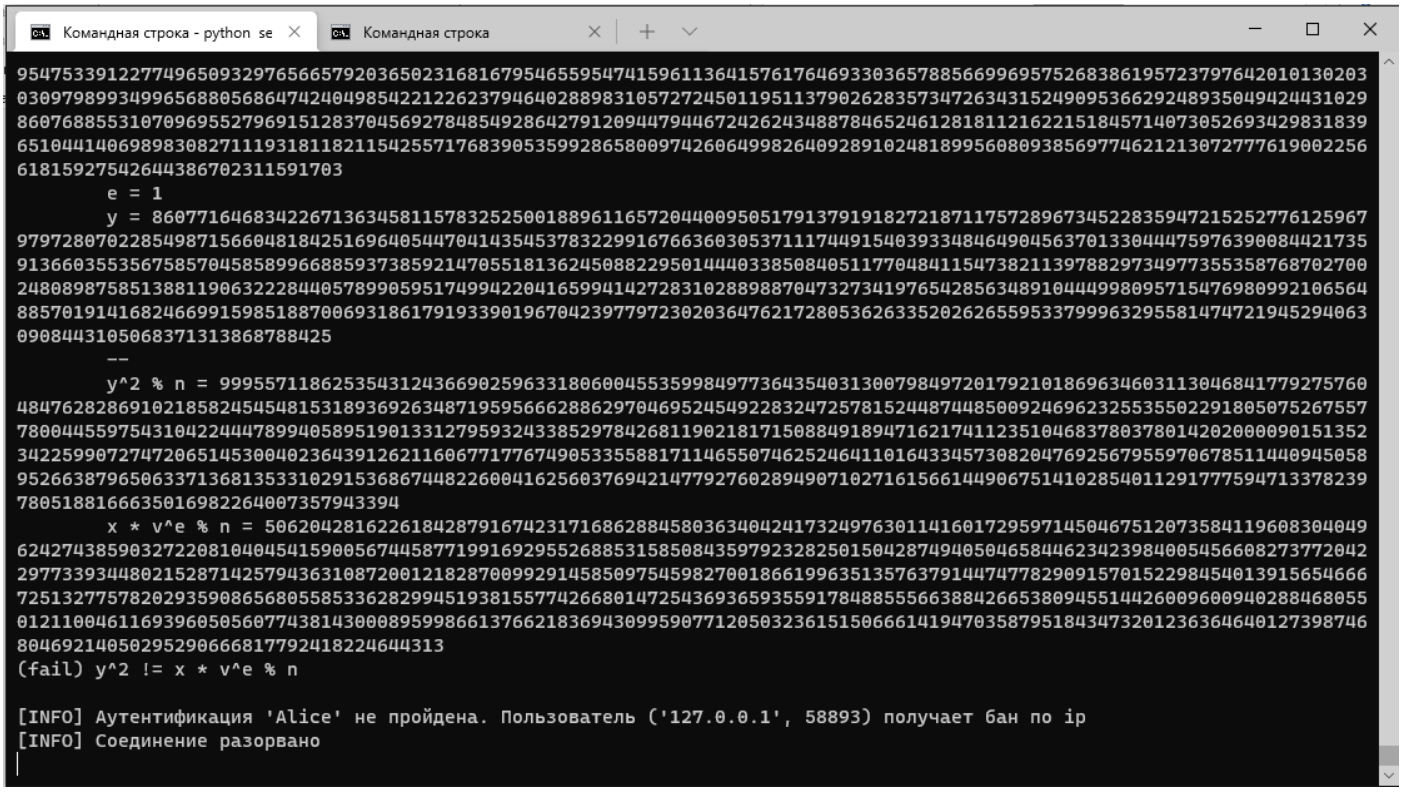
Результаты


Командная строка - python se
Командная строка
C:\Users\User\Desktop\ЗИ\RGR\src>python server.py
p = 104908074358786932409223619070694781545548552393663815058454335281535463650174985168751464269780251717407289
114346927649405014168602219525494856886179661938210416713584268949076053291033047555384027521929414477923915901075244794
690852796011254926796269007020019482182385027578437438692510907699426282941366983
q = 167001018650637090222921427714875522066500295520321291928330919166260223558669265877919090616664565208552555
662582755208210073527694779793168239913886595632745682308971422503665665324991029789555395661761104555004987881007999935
040276450281725588219509555067490455229116558554110808058616034860660141535000633
self.n = 175197552825941991991285403220932945446652972266429249605664525756112904129458056388688620797741932624548
481495472424926687736888002617545270152328935366417229421964010235956065357303729546890886151054856019392455623521241318
994048979165520255582863355921774267706470351520822991574645997028876624961302450964192421264642327577036956407192978395
463839660680167183413603490710932517408969049644621352843539100176555314058903033246694675565486619735966113309808490176
271306838444809995265761164607260106336557657816890398619211051981834452060012479162089382981505907090617810339692420267
205528935602585943199304079030023 9
****
[INFO] Загружена база с пользователями
[INFO] Сервер запущен

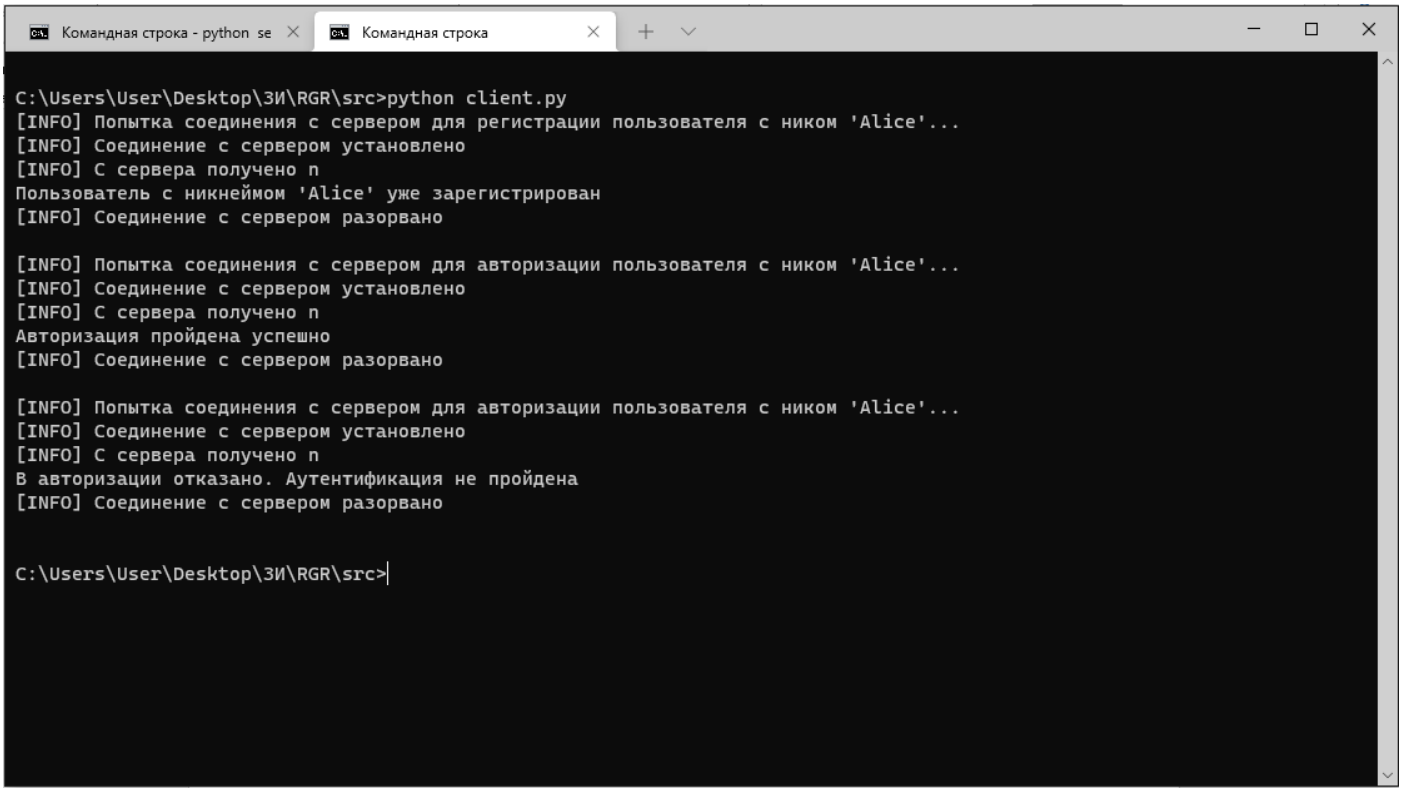

Командная строка - python se
Командная строка
[INFO] Соединение разорвано
[INFO] Установлено новое соединение: ('127.0.0.1', 58891)
[INFO] Пользователю ('127.0.0.1', 58891) отправлено n
[ERROR] Никнейм 'Alice' уже зарегистрирован
[INFO] Соединение разорвано
[INFO] Установлено новое соединение: ('127.0.0.1', 58892)
[INFO] Пользователю ('127.0.0.1', 58892) отправлено n
[INFO] Пользователь ('127.0.0.1', 58892) пытается авторизироваться под никнеймом 'Alice'
[INFO] Аутентификация:
1.
x = 602129939163185014454482058184724025991593631619578149148135953843980862961103605838873592780016379034886926
382332412657243121443685568113072839050338153441975527618421788045493431334836934421234993915223647524607381627141552730
741918132198095179690084410608221601685018836387201704926859867774966020337612931786400077644927366793512190595736067816
521611090370867476920654599754073233590818265990361890901296311579760925425775993222158467682107737370680193398602395103
851979330614987552449767601374785636163730394753619703219742105160730496993481030864115257580106206095905995440204208131
453794858922927070560400032 9
e = 1
y = 958721570049226835370442144639013871546250144035532305782322936501314772957199598718150986249672424272903506
102837590144914349197067741581343706291659502659041931888222547017023606261288810475809539776122635052542313879128764476
813668606477754511910673169796927274794349637562141041755758720112715437510343435115320258390507403496433050107852036528
916648870156673060717231534708651323958548698544817454061774967102758050592153694954217775590489496961846055708381271987
194248747349944976583139768584487707386673698201438210457853042369881381299782475873867041922480408492735067202365939862
688926205236773616919181521 9
--
y^2 % n = 103353776821895140592461322740103840990246182109920470012284449696936896093884474483469198676269557496
425131725230038012026021690925716290167820399147548207940823971117797072407727450184131334156822462539151784827138191410
335859886625529135562540713121520215790902021230897275046123725824778509922332268753481764596734802542577416193787109555
853062542023569314980086269894994174585882987275725984945048988393308907721803497008381509474157235612350699285046122006


Командная строка - python se
Командная строка
x = 506691423098641632862857007595822129968149614462741880209016428007853236570488730703540231164284300970896615
672657409493600541941445010856098390143981701531307012895096176940314367384295748894998487038604391244446006781111958721
445054160390034509164495562407794821536369214828728918372155995415976247623081989380223452446288713878512753704107565691
809646387958122992890602001090431502772698285438776997457926295261852994751062913694853552297015168111437604528201610190
623999751554595002895609038683720786492012480319373130332039648315187053109801175874598320160163379054073669538862529790
990507187683136125354260865
e = 1
y = 187312764765781462638416179244473753796863482675522367483196219488320453087897052130880685922090176819768895
535965254418939946676890824689134596636852708865879476494576561305308751010345829725505529715176453173950152164243652739
532164248883918172883212715880940514641528739440746884177550651216643549323843906479057093661057704992273000637052100713
797590920187669993280642830548495729915131904473794206528649760436400914264249199268087765523477850382929694301268140357
498722000308493461498921827086210223149359049258877105871370221333803031121440669572065508723705143284183929466651061364
228033824851457953666728717 6
--
y^2 % n = 110795456619053574211536188428390955795589139129450655141992902223623484580971934744068393893960033312
843405834149740180030998472330555325907987605732197355485144035559822510323174601079366051648435508041110313307658780495
673841595467304869794317717007185820956296879782513776274315142915923213492138750631985792537649810904212826751110091076
653809021799775873789551872828018665449487824838913100930195852774943729316620211534095187081045630306866963459737574558
306304341579768883004651586664090766776783477510682908023728229954391592340876197069203426400533706690743486587017999469
090181468250207944041505222169157 27
x * v^e % n = 110795456619053574211536188428390955795589139129450655141992902223623484580971934744068393893960033
331284340583414974018003099847233055532590798760573219735548514403555982251032317460107936605164843550804111031330765878
049567384159546730486979431771700718582095629687978251377627431514291592321349213875063198579253764981090421282675111009
107665380902179977587378955187282801866544948782483891310093019585277494372931662021153409518708104563030686696345973757
455830630434157976888300465158666409076677678347751068290802372822995439159234087619706920342640053370669074348658701799
946909018146825020794404150522216915727
(success) y^2 == x * v^e % n

[INFO] Аутентификация 'Alice' пройдена успешно
[INFO] Соединение разорвано


Командная строка - python se
Командная строка
954753391227749650932976566579203650231681679546559547415961136415761764693303657885669969575268386195723797642010130203
030979899349965688056864742404985422122623794640288983105727245011951137902628357347263431524909536629248935049424431029
860768855310709695527969151283704569278485492864279120944794467242624348878465246128181121622151845714073052693429831839
651044140698983082711193181182115425571768390535992865800974260649982640928910248189956080938569774621213072777619002256
618159275426443867023115917 03
e = 1
y = 860771646834226713634581157832525001889611657204400950517913791918272187117572896734522835947215252776125967
979728070228549871566048184251696405447041435453783229916766360305371117449154039334846490456370133044475976390084421735
913660355356758570458589966885937385921470551813624508822950144403385084051177048411547382113978829734977355358768702700
248089875851388119063222844057899059517499422041659941427283102889887047327341976542856348910444998095715476980992106564
885701914168246699159851887006931861791933901967042397797230203647621728053626335202626559533799963295581474721945294063
090844310506837131386878842 5
--
y^2 % n = 999557118625354312436690259633180600455359984977364354031300798497201792101869634603113046841779275760
484762828691021858245454815318936926348719595666288629704695245492283247257815244874485009246962325535502291805075267557
780044559754310422444789940589519013312795932433852978426811902181715088491894716217411235104683780378014202000090151352
342259907274720651453004023643912621160677177674905335588171146550746252464110164334573082047692567955970678511440945058
952663879650633713681353310291536867448226004162560376942147792760289490710271615661449067514102854011291777594713378239
780518816663501698226400735794339 4
x * v^e % n = 506204281622618428791674231716862884580363404241732497630114160172959714504675120735841196083040496
242743859032722081040454159005674458771991692955268853158508435979232825015042874940504658446234239840054566082737720422
977339344802152871425794363108720012182870099291458509754598270018661996351357637914474778290915701522984540134915654666
725132775782029359086568055853362829945193815577426680147254369365935591784885556638842665380945514426009600940288468055
012110046116939605056077438143000895998661376621836943099590771205032361515066614194703587951843473201236364640127398746
804692140502952906668177924182246443 13
(fail) y^2 != x * v^e % n

[INFO] Аутентификация 'Alice' не пройдена. Пользователь ('127.0.0.1', 58893) получает бан по ip
[INFO] Соединение разорвано


Командная строка - python se
Командная строка
C:\Users\User\Desktop\ЗИ\RGR\src>python client.py
[INFO] Попытка соединения с сервером для регистрации пользователя с ником 'Alice'...
[INFO] Соединение с сервером установлено
[INFO] С сервера получено n
Пользователь с никнеймом 'Alice' уже зарегистрирован
[INFO] Соединение с сервером разорвано
[INFO] Попытка соединения с сервером для авторизации пользователя с ником 'Alice'...
[INFO] Соединение с сервером установлено
[INFO] С сервера получено n
Авторизация пройдена успешно
[INFO] Соединение с сервером разорвано
[INFO] Попытка соединения с сервером для авторизации пользователя с ником 'Alice'...
[INFO] Соединение с сервером установлено
[INFO] С сервера получено n
В авторизации отказано. Аутентификация не пройдена
[INFO] Соединение с сервером разорвано
C:\Users\User\Desktop\ЗИ\RGR\src>

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
```