Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств
вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Защита информации

Расчётно-графическая работа

Вариант №3 (Протокол Фиата-Шамира)

Выполнил:

студент гр.ИП-213 Дмитриев Антон Александрович
ФИО студента

«__» _____ 2025 г.

Проверил:

Преподаватель Дьячкова Марина Сергеевна
ФИО преподавателя

«__» _____ 2025 г. Оценка_____

Новосибирск 2025 г.

1. Краткая теория

Основные действия (этапы каждого раунда):

Следующие действия последовательно и независимо выполняются t раз.

А доказывает В знание s в течение t раундов (аккредитаций).

В считает знание доказанным, если все t раундов прошли успешно.

1. А выбирает случайное r, такое, что $\boldsymbol{r \in 1, N - 1}$ и отсылает $\mathbf{x} = \boldsymbol{r\text{^}2 \ mod \ N}$ стороне В (доказательство).

2. В случайно выбирает бит e (e = 0 или e = 1) и отсылает его А (вызов).

3. А вычисляет $\boldsymbol{y = rs\text{^}e \ mod \ N}$ и отправляет его обратно к В.

Если $\mathbf{e = 0}$, то $\boldsymbol{y = r}$, иначе (при $\mathbf{e = 1}$) $\boldsymbol{y = rs \ mod \ N}$.

4. Если $\mathbf{y = 0}$, то В отвергает доказательство (т.е. А не удалось доказать знание

s). Иначе сторона В проверяет, действительно ли $\boldsymbol{y\text{^}2 = xv\text{^}e \ mod \ N}$ и если это так, топроисходит переход к следующему раунду протокола.

Выбор e из множества 0, 1 предполагает, что если сторона А действительно знает секрет, то она всегда сможет правильно ответить, вне зависимости от выбранного e.

2. Описание функций

def extended_gcd(a, b) - Нахождение НОД

def fast_pow(a, x, p): - Быстрое возведение в степень

def ferm_test(n): - Проверка на простоту

def generate_N(self): - Генерация модуля N = p*q

def register_user(self, username, v): - Регистрация пользователя с открытым ключом v

def save_users(self): - Сохранение пользователей в файл

def load_users(self): - Загрузка пользователей из файла

def get_user(self, username): - Получение информации о пользователе

def start_authentication(self, username): - Начало аутентификации пользователя

def receive_x(self, x): - Получение доказательства x от клиента

def receive_y(self, y): - Получение ответа y от клиента и проверка

def is_authenticated(self): - Проверка успешности всех раундов

def generate_keys(self, N, username): - Генерация секретного и открытого ключей

def load_keys(self, N, s, username): - Загрузка существующих ключей

def start_authentication(self, server_params): - Начало аутентификации с сервером

def generate_x(self): - Генерация доказательства x = r^2 mod N

def compute_y(self, e): - Вычисление ответа y = r * s^e mod N

def get_round_info(self, round_num): - Получение информации о раунде

def simulate_protocol():- Симуляция полного протокола Фиата-Шамира

def interactive_mode():- Интерактивный режим работы

3. Тесты

```
--- Раунд 20/20 ---
КЛИЕНТ: Раунд 20:
  Выбрано r = 2288855384573978385550851875745938238992280926462023023911896370390730356595663033521606301177279505039127965581915891323391508484118546952006745848713357662653119997764879757213026676014584603067014825540253387944973071900740710080653562250637859473283512762151489264626480736515340910444318845766059271650 4
  Отправляем x = r^2 mod N = 852474399158594391519018125671489449402228087017831775785464416346539405359387467932983935102722883424614607672073203722950085821178793252093911587294236986786314035149218841163533039312030209722935874680887619532069026914377772123883227073872496695443868706466155491299871947908712531720495097002181679049970
СЕРВЕР: Получено x = 852474399158594391519018125671489449402228087017831775785464416346539405359387467932983935102722883424614607672073203722950085821178793252093911587294236986786314035149218841163533039312030209722935874680887619532069026914377772123883227073872496695443868706466155491299871947908712531720495097002181679049970
СЕРВЕР: Сгенерирован вызов e = 0
КЛИЕНТ: Получен вызов e = 1
КЛИЕНТ: Вычисляем y = r * s^1 mod N = 400471626694332197041880789346206128886162129678038813128092173815675722211986983444687750301590013588145209280600591823812314244798679968926665071459204723683110879866477670930988397318795365996843418690390327477167355464017969980749831056026991613728693974925827701190704878863757010679599279833118377585820
СЕРВЕР: Проверка раунда 20:
  y^2 mod N = 136435604191925895636573252127983943354400036919006026865220491228480492846470423458267774532544078734856847461471459148146432339946833034626948426331616622460666256733109690082462120846132141423759562829707854332967820815292693433774844151604089941154795272714771158939543476206754153026361182779398600598316
  x * v^0 mod N = 852474399158594391519018125671489449402228087017831775785464416346539405359387467932983935102722883424614607672073203722950085821178793252093911587294236986786314035149218841163533039312030209722935874680887619532069026914377772123883227073872496695443868706466155491299871947908712531720495097002181679049970
  ✗ Раунд 20 не пройден!
✗ Раунд 20 не пройден!

СЕРВЕР: Аутентификация не удалась!
СЕРВЕР: Только 19 из 20 раундов пройдены
СЕРВЕР: Пользователи сохранены в users.json

============================================================
✗ АУТЕНТИФИКАЦИЯ НЕ УДАЛАСЬ!
============================================================
```

Рисунок 1. Попытка входа с обманом

```
--- Раунд 20/20 ---
КЛИЕНТ: Раунд 20:
  Выбрано r = 5510982211462202294873281945113032126379755785192873315931555685341101343076695099293420367882025560566436601567290454165833710465791613072696084922541181486468379759046991893456327119203288865822465570924319187956877168044922090003440822522684645737353596375315152612693102308740848942307978113353836898349524
  Отправляем x = r^2 mod N = 167236319224676316721269926585210993065327456537232677868648172575309926678076838239098281368650052033014665812997258210215939197331311932800685702011144658088435153813345069557716748711603057696439766623789043754117878012591761410134023683390358187831930735289621242278156612017427180947152245908566829917491
СЕРВЕР: Получено x = 167236319224676316721269926585210993065327456537232677868648172575309926678076838239098281368650052033014665812997258210215939197331311932800685702011144658088435153813345069557716748711603057696439766623789043754117878012591761410134023683390358187831930735289621242278156612017427180947152245908566829917491
СЕРВЕР: Сгенерирован вызов e = 0
КЛИЕНТ: Получен вызов e = 0
КЛИЕНТ: Вычисляем y = r * s^0 mod N = 551098221146220229487328194511303212637975578519287331593155568534110134307669509929342036788202556056643660158 4
  y^2 mod N = 167236319224676316721269926585210993065327456537232677868648172575309926678076838239098281368650052033014665812997258210215939197331311932800685702011144658088435153813345069557716748711603057696439766623789043754117878012591761410134023683390358187831930735289621242278156612017427180947152245908566829917491
  x * v^0 mod N = 167236319224676316721269926585210993065327456537232677868648172575309926678076838239098281368650052033014665812997258210215939197331311932800685702011144658088435153813345069557716748711603057696439766623789043754117878012591761410134023683390358187831930735289621242278156612017427180947152245908566829917491
  ✓ Раунд 20 пройден успешно!
  Успешных раундов: 20/20

СЕРВЕР: Аутентификация успешна!
СЕРВЕР: Все 20 раундов пройдены
СЕРВЕР: Добро пожаловать, t!
СЕРВЕР: Успешных входов: 1
СЕРВЕР: Пользователи сохранены в users.json

✓ Добро пожаловать, t!
```

Рисунок 2. Успешная попытка входа

4. Листинг кода

```
import random
import json
import os
from pathlib import Path


def extended_gcd(a, b):
    """Нахождение НОД"""
    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 fast_pow(a, x, p):
    """Быстрое возведение в степень"""
    y = 1
    a = a % p

    while x > 0:
        if x & 1:
            y = (y * a) % p
        a = (a * a) % p
        x >>= 1

    return y
```

```
def ferm_test(n):
    """Проверка на простоту"""
    k = 50

    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False

    for _ in range(k):
        a = random.randint(2, n - 2)
        if fast_pow(a, n - 1, n) != 1:
            return False

    return True

class FiatShamirServer:
    """Серверная часть протокола Фиата-Шамира"""

    def __init__(self):
        self.N = 0 # Модуль N = p*q
        self.p = 0 # Секретное простое p
        self.q = 0 # Секретное простое q
        self.users_file = "users.json"
        self.users = {}
        self.current_session = {}
```

```
def generate_N(self):
    """Генерация модуля N = p*q"""
    print("СЕРВЕР: Генерация модуля N...")
    # Генерируем простые числа p и q
    while True:
        self.p = random.randint(2**511, 2**512)  # 512-битные простые
        if ferm_test(self.p):
            break

    while True:
        self.q = random.randint(2**511, 2**512)
        if ferm_test(self.q) and self.q != self.p:
            break

    # Вычисляем параметры
    self.N = self.p * self.q

    print(f"СЕРВЕР: p = {self.p}")
    print(f"СЕРВЕР: q = {self.q}")
    print(f"СЕРВЕР: N = p*q = {self.N}")
    print(f"СЕРВЕР: Битность N: {self.N.bit_length()} бит")

    return self.N

def register_user(self, username, v):
    """Регистрация пользователя с открытым ключом v"""
    if username in self.users:
        print(f"СЕРВЕР: Пользователь {username} уже существует!")
        return False
```

```
    # Проверяем, что v в правильном диапазоне
    if v <= 1 or v >= self.N:
        print(f"СЕРВЕР: v должно быть в диапазоне (1, N-1)")
        return False

    self.users[username] = {
        'username': username,
        'v': v,  # Открытый ключ: v = s^2 mod N
        'login_attempts': 0
    }

    self.save_users()
    print(f"СЕРВЕР: Пользователь {username} зарегистрирован с v={v}")
    return True

def save_users(self):
    """Сохранение пользователей в файл"""
    with open(self.users_file, 'w') as f:
        json.dump(self.users, f, indent=2)
    print(f"СЕРВЕР: Пользователи сохранены в {self.users_file}")

def load_users(self):
    """Загрузка пользователей из файла"""
    if os.path.exists(self.users_file):
        with open(self.users_file, 'r') as f:
            self.users = json.load(f)
        print(f"СЕРВЕР: Загружено {len(self.users)} пользователей")
        return True
    return False
```

```
def get_user(self, username):
    """Получение информации о пользователе"""
    return self.users.get(username)

def start_authentication(self, username):
    """Начало аутентификации пользователя"""
    if username not in self.users:
        print(f"СЕРВЕР: Пользователь {username} не найден!")
        return None

    # Сбрасываем текущую сессию
    self.current_session = {
        'username': username,
        'v': self.users[username]['v'],
        'rounds': 0,
        'successful_rounds': 0,
        'total_rounds': 20,  # Количество раундов t
        'current_round': 0,
        'x_received': None,
        'e_sent': None,
        'y_received': None
    }

    print(f"СЕРВЕР: Начало аутентификации для {username}")
    print(f"СЕРВЕР: Открытый ключ v={self.current_session['v']}")
    print(f"СЕРВЕР: Будет проведено {self.current_session['total_rounds']} раундов")

    return {
        'N': self.N,
        'v': self.current_session['v'],
```

```
        'total_rounds': self.current_session['total_rounds']
    }

def receive_x(self, x):
    """Получение доказательства x от клиента"""
    # Проверяем, что x в правильном диапазоне
    if x <= 0 or x >= self.N:
        print(f"СЕРВЕР: x должно быть в диапазоне (1, N-1)")
        return False

    self.current_session['x_received'] = x
    print(f"СЕРВЕР: Получено x = {x}")

    # Генерируем случайный бит e (0 или 1)
    e = random.randint(0, 1)
    self.current_session['e_sent'] = e
    print(f"СЕРВЕР: Сгенерирован вызов e = {e}")

    return e

def receive_y(self, y):
    """Получение ответа y от клиента и проверка"""
    if y == 0:
        print("СЕРВЕР: y = 0! Доказательство отвергнуто.")
        return False

    self.current_session['y_received'] = y
    self.current_session['current_round'] += 1

    x = self.current_session['x_received']
```

```
        v = self.current_session['v']
        e = self.current_session['e_sent']

        # Проверяем: y^2 ≡ x * v^e mod N
        left_side = (y * y) % self.N
        right_side = (x * fast_pow(v, e, self.N)) % self.N

        print(f"СЕРВЕР: Проверка раунда {self.current_session['current_round']}:")
        print(f"  y^2 mod N = {left_side}")
        print(f"  x * v^{e} mod N = {right_side}")

        if left_side == right_side:
            self.current_session['successful_rounds'] += 1
            print(f"  ✓ Раунд {self.current_session['current_round']} пройден успешно!")
            print(f"  Успешных раундов:
{self.current_session['successful_rounds']}/{self.current_session['total_rounds']}")
            return True
        else:
            print(f"  ✗ Раунд {self.current_session['current_round']} не пройден!")
            return False

    def is_authenticated(self):
        """Проверка успешности всех раундов"""
        if self.current_session['current_round'] < self.current_session['total_rounds']:
            return False

        success = self.current_session['successful_rounds'] == self.current_session['total_rounds']

        if success:
            print(f"\nСЕРВЕР: Аутентификация успешна!")
```

```
            print(f"СЕРВЕР: Все {self.current_session['total_rounds']} раундов пройдены")
            print(f"СЕРВЕР: Добро пожаловать, {self.current_session['username']}!")
        else:
            print(f"\nСЕРВЕР: Аутентификация не удалась!")
            print(f"СЕРВЕР: Только {self.current_session['successful_rounds']} из
{self.current_session['total_rounds']} раундов пройдены")

        # Обновляем статистику пользователя
        if self.current_session['username'] in self.users:
            self.users[self.current_session['username']]['login_attempts'] += 1
            if success:
                print(f"СЕРВЕР: Успешных входов:
{self.users[self.current_session['username']]['login_attempts']}")

        self.save_users()
        return success

class FiatShamirClient:
    """Клиентская часть протокола Фиата-Шамира"""

    def __init__(self):
        self.N = 0  # Модуль от сервера
        self.s = 0  # Секретный ключ (известен только клиенту)
        self.v = 0  # Открытый ключ (хранится на сервере)
        self.username = ""
        self.current_session = {}

    def generate_keys(self, N, username):
        """Генерация секретного и открытого ключей"""
        self.N = N
        self.username = username
```

```
    # Выбираем s, взаимно простое с N
    while True:
        self.s = random.randint(2, N - 2)
        if extended_gcd(self.s, N)[0] == 1:
            break

    # Вычисляем v = s^2 mod N
    self.v = (self.s * self.s) % N

    print(f"КЛИЕНТ: Сгенерированы ключи для {username}:")
    print(f"  Секретный ключ s = {self.s}")
    print(f"  Открытый ключ v = s^2 mod N = {self.v}")
    print(f"  Проверка: gcd(s, N) = {extended_gcd(self.s, N)[0]} (должно быть 1)")

    return self.v

def load_keys(self, N, s, username):
    """Загрузка существующих ключей"""
    self.N = N
    self.s = s
    self.username = username
    self.v = (s * s) % N

    print(f"КЛИЕНТ: Загружены ключи для {username}:")
    print(f"  s = {s}")
    print(f"  v = {self.v}")

def start_authentication(self, server_params):
    """Начало аутентификации с сервером"""
```

```
    self.N = server_params['N']
    self.v = server_params['v']
    total_rounds = server_params['total_rounds']

    self.current_session = {
        'total_rounds': total_rounds,
        'current_round': 0,
        'successful_rounds': 0,
        'r_values': [],
        'x_values': [],
        'e_values': [],
        'y_values': []
    }

    print(f"КЛИЕНТ: Начало аутентификации для {self.username}")
    print(f"КЛИЕНТ: N = {self.N}")
    print(f"КЛИЕНТ: v = {self.v}")
    print(f"КЛИЕНТ: Будет проведено {total_rounds} раундов")

def generate_x(self):
    """Генерация доказательства x = r^2 mod N"""
    # Выбираем случайное r ∈ [1, N-1]
    r = random.randint(1, self.N - 1)

    # Вычисляем x = r^2 mod N
    x = (r * r) % self.N

    # Сохраняем для текущего раунда
    self.current_session['r_values'].append(r)
    self.current_session['x_values'].append(x)
```

```
        self.current_session['current_round'] += 1

        print(f"КЛИЕНТ: Раунд {self.current_session['current_round']}:")
        print(f"  Выбрано r = {r}")
        print(f"  Отправляем x = r^2 mod N = {x}")

        return x

    def compute_y(self, e):
        """Вычисление ответа y = r * s^e mod N"""
        current_round = self.current_session['current_round'] - 1
        r = self.current_session['r_values'][current_round]

        # Сохраняем вызов e
        self.current_session['e_values'].append(e)

        # Вычисляем y = r * s^e mod N
        if e == 0:
            y = r % self.N
        else:  # e == 1
            y = (r * self.s) % self.N

        self.current_session['y_values'].append(y)

        print(f"КЛИЕНТ: Получен вызов e = {e}")
        print(f"КЛИЕНТ: Вычисляем y = r * s^{e} mod N = {y}")

        return y

    def get_round_info(self, round_num):
```

```
        """Получение информации о раунде"""
        if round_num < 0 or round_num >= len(self.current_session['r_values']):
            return None

        return {
            'r': self.current_session['r_values'][round_num],
            'x': self.current_session['x_values'][round_num],
            'e': self.current_session['e_values'][round_num],
            'y': self.current_session['y_values'][round_num]
        }

def simulate_protocol():
    """Симуляция полного протокола Фиата-Шамира"""
    print("="*60)
    print("СИМУЛЯЦИЯ ПРОТОКОЛА ФИАТА-ШАМИРА")
    print("="*60)

    # Создаем сервер
    server = FiatShamirServer()
    server.load_users()

    # Генерируем модуль N
    N = server.generate_N()

    # Создаем клиента
    client = FiatShamirClient()

    # Регистрация нового пользователя
    username = input("\nВведите имя пользователя для регистрации: ")
```

```
# Клиент генерирует ключи
v = client.generate_keys(N, username)

# Сервер регистрирует пользователя
if server.register_user(username, v):
    print(f"\n✓ Пользователь {username} успешно зарегистрирован!")
else:
    print(f"\n✗ Ошибка регистрации пользователя {username}")
    return

# Аутентификация
print("\n" + "="*60)
print("НАЧАЛО АУТЕНТИФИКАЦИИ")
print("="*60)

# Клиент начинает аутентификацию
server_params = server.start_authentication(username)
client.start_authentication(server_params)

# Выполняем t раундов
t = server_params['total_rounds']

for round_num in range(t):
    print(f"\n--- Раунд {round_num + 1}/{t} ---")

    # Шаг 1: Клиент выбирает r и отправляет x = r^2 mod N
    x = client.generate_x()

    # Шаг 2: Сервер получает x и отправляет вызов e
    e = server.receive_x(x)
```

```
    if round_num == 19:
        if e == 0:
            e = 1
        else:
            e = 0

    # Шаг 3: Клиент вычисляет y = r * s^e mod N
    y = client.compute_y(e)

    # Шаг 4: Сервер проверяет y
    if not server.receive_y(y):
        print(f"✗ Раунд {round_num + 1} не пройден!")
        break

# Проверяем успешность аутентификации
if server.is_authenticated():
    print("\n" + "="*60)
    print("✓ АУТЕНТИФИКАЦИЯ ПРОЙДЕНА УСПЕШНО!")
    print("="*60)

    # Показываем детали последнего раунда
    if client.current_session['current_round'] > 0:
        last_round = client.get_round_info(client.current_session['current_round'] - 1)
        if last_round:
            print("\nДетали последнего раунда:")
            print(f"  r = {last_round['r']}")
            print(f"  x = r^2 mod N = {last_round['x']}")
            print(f"  e = {last_round['e']}")
            print(f"  y = r * s^{last_round['e']} mod N = {last_round['y']}")
```

```
            # Проверяем вручную
            y_sq = (last_round['y'] * last_round['y']) % N
            x_v_e = (last_round['x'] * fast_pow(v, last_round['e'], N)) % N
            print(f"  Проверка: y^2 = {y_sq}, x * v^e = {x_v_e}")
            print(f"  Совпадают: {y_sq == x_v_e}")
    else:
        print("\n" + "="*60)
        print("✗ АУТЕНТИФИКАЦИЯ НЕ УДАЛАСЬ!")
        print("="*60)

def interactive_mode():
    """Интерактивный режим работы"""
    server = FiatShamirServer()
    server.load_users()

    if not server.users:
        print("СЕРВЕР: Генерация нового модуля N...")
        server.generate_N()

    print(f"\nСЕРВЕР: Модуль N = {server.N}")
    print(f"СЕРВЕР: Зарегистрированных пользователей: {len(server.users)}")

    while True:
        print("\n" + "="*60)
        print("МЕНЮ ПРОТОКОЛА ФИАТА-ШАМИРА")
        print("="*60)
        print("1. Зарегистрировать нового пользователя")
        print("2. Аутентифицировать существующего пользователя")
        print("3. Показать всех пользователей")
```

```
print("4. Симуляция полного протокола")
print("5. Выход")

choice = input("\nВыберите действие: ")

if choice == '1':
    # Регистрация нового пользователя
    username = input("Введите имя пользователя: ")

    if username in server.users:
        print(f'Пользователь {username} уже существует!")
        continue

    print(f"\nГенерация ключей для {username}...")

    # Создаем клиента
    client = FiatShamirClient()

    # Клиент генерирует ключи
    v = client.generate_keys(server.N, username)

    # Показываем ключи пользователю
    print(f"\nКлючи пользователя {username}:")
    print(f"  Секретный ключ s = {client.s} (НИКОМУ НЕ ПОКАЗЫВАЙТЕ!)")
    print(f"  Открытый ключ v = {v} (будет отправлен на сервер)")

    # Регистрируем на сервере
    if server.register_user(username, v):
        print(f"\n✓ Пользователь {username} успешно зарегистрирован!")
        print("ВАЖНО: Сохраните свой секретный ключ s для последующих входов!")
```

```
elif choice == '2':
    # Аутентификация
    username = input("Введите имя пользователя: ")

    if username not in server.users:
        print(f"Пользователь {username} не найден!")
        continue

    print(f"\nАутентификация пользователя {username}...")

    # Запрашиваем секретный ключ
    try:
        s = int(input("Введите ваш секретный ключ s: "))
    except ValueError:
        print("✗ Неверный формат ключа!")
        continue

    # Создаем клиента с загруженным ключом
    client = FiatShamirClient()
    client.load_keys(server.N, s, username)

    # Проверяем, что v совпадает
    if client.v != server.users[username]['v']:
        print("✗ Неверный секретный ключ! v не совпадает.")
        continue

    # Начинаем аутентификацию
    server_params = server.start_authentication(username)
    client.start_authentication(server_params)
```

```
    t = server_params['total_rounds']
    all_rounds_successful = True

    # Выполняем раунды
    for round_num in range(t):
        print(f"\n--- Раунд {round_num + 1}/{t} ---")

        # Клиент отправляет x
        x = client.generate_x()

        # Сервер отправляет e
        e = server.receive_x(x)

        # Клиент вычисляет y
        y = client.compute_y(e)

        # Сервер проверяет
        if not server.receive_y(y):
            all_rounds_successful = False
            print(f"✗ Раунд {round_num + 1} не пройден!")
            break

    # Проверяем результат
    if all_rounds_successful and server.is_authenticated():
        print(f"\n✓ Добро пожаловать, {username}!")
    else:
        print(f"\n✗ Аутентификация не удалась!")

elif choice == '3':
```

```
            # Показать всех пользователей
            print(f"\nЗарегистрированные пользователи ({len(server.users)}):")
            for username, data in server.users.items():
                print(f"  {username}: v={data['v']}, входов: {data.get('login_attempts', 0)}")

        elif choice == '4':
            # Симуляция полного протокола
            simulate_protocol()

        elif choice == '5':
            print("Выход из программы")
            break

        else:
            print("✗ Неверный выбор!")

def main():
    print("="*60)
    print("ПРОТОКОЛ ДОКАЗАТЕЛЬСТВА С НУЛЕВЫМ ЗНАНИЕМ ФИАТА-ШАМИРА")
    print("="*60)
    print("Реализация клиент-серверной аутентификации")
    print("Стойкость основана на сложности извлечения квадратного корня по модулю N")

    interactive_mode()

if __name__ == "__main__":
    main()
```