

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

Схемы аутентификации
ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Яхина Шамяля Илдусовича

Преподаватель
аспирант

_____ Р. А. Фарахутдинов
подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Теоретические сведения	4
1.1 Описание алгоритма упрощенной схемы идентификации Фейге- Фиата-Шамира	4
2 Практическая реализация	6
2.1 Описание программы	6
2.2 Тестирование программы.....	6

ВВЕДЕНИЕ

Цель работы - реализация упрощенной схемы идентификации Фейге-Фиата-Шамира.

1 Теоретические сведения

Генерация общих параметров. Доверенный центр T (Трент) публикует большое число $n = p * q$, где p, q — большие простые числа, которые держатся в секрете. Также выбираются целые числа k и t — параметры безопасности.

Генерация открытого и закрытого ключей. Для генерации открытого и закрытого ключей Пегги доверенный арбитр выбирает число v , являющееся квадратичным остатком $\bmod n$. Другими словами, выбирается v так, чтобы уравнение $x^2 \equiv v \pmod{n}$ имело решение, и существовало $v^{-1} \bmod n$. Это v и будет открытым ключом Пегги. Затем вычисляется наименьшее s , для которого $s \equiv \sqrt{v^{-1}} \pmod{n}$. Это будет закрытый ключ Пегги.

1.1 Описание алгоритма упрощенной схемы идентификации Фейге-Фиата-Шамира

Алгоритм упрощенной схемы идентификации Фейге-Фиата-Шамира.

Вход: Простое число n , по модулю которого производятся вычисления, и g - порождающий элемент группы.

Выход: Сеансовый ключ K .

Алиса доказывает своё знание секрета s Бобу в течение t раундов, не раскрывая при этом ни одного бита самого секрета. Действие протокола в рамках одного раунда (одной аккредитации):

1. Пегги передает Виктору x , где $x = (-r)^2 \bmod n$, r - новое случайное число Вегги, $1 \leq r \leq n - 1$;
2. Виктор передает Пегги случайный бит b ;
3. Если $b = 0$, то Пегги посылает Виктору r . Если $b = 1$, то Пегги посылает Виктору $y = r * s \bmod n$;
4. Если $b = 0$, Виктор проверяет, что $x = (-r)^2 \bmod n$, убеждаясь, что Пегги знает значение \sqrt{x} . Если $b = 1$, Виктор проверяет, что $x = y^2 * v \bmod n$, убеждаясь, что Пегги знает значение $\sqrt{(v^{-1})}$.

Это один этап протокола, называемый аккредитацией. Пегги и Виктор повторяют этот протокол t раз, пока Виктор не убедится, что Пегги знает s . Если Пегги не знает s , она может подобрать r так, что она сможет обмануть Виктора, если он пошлет ей 0, или она может подобрать r так, что она сможет обмануть Виктора, если он пошлет ей 1. Она не может сделать одновременно и то, и другое. Вероятность, что ей удастся обмануть Виктора один раз, равна 50

процентам . Вероятность, что ей удастся обмануть его t раз, равна $\frac{1}{2^t}$.

Виктор может попробовать вскрыть протокол, выдавая себя за Пегги . Он может начать выполнение протокола с другим контролером, Валерией. На шаге 1 вместо выбора случайного r ему останется просто использовать значение r , которое Пегги использовало в прошлый раз. Однако, вероятность того, что Валерия на шаге 2 выберет то же значение b , которое Виктор использовал в протоколе с Пегги, равна $\frac{1}{2}$.

Чтобы этот протокол работал, Пегги никогда не должна использовать r повторно. В противном случае, если Виктор на шаге 2 пошлет Пегги другой случайный бит, то он получит оба ответа Пегги. Тогда даже по одному из них он сможет вычислить s , и для Пегги все закончится.

2 Практическая реализация

2.1 Описание программы

Все действия алгоритма в функции *FFS_main*.

Генерация больших простых чисел p, q и вычисление по ним n происходит в функции *generate_n*, где происходит вызов функций *generateRandomPrime* для генерации простого случайного числа в заданном диапазоне, а для проверки числа на простоту с помощью теста Миллера-Рабина используется функция *isPrimeMillerRabin*.

Публичный ключ генерируется в функции *generatePublicKey*, в данной функции используется функция *jacobi_symbol* для вычисления символа Якоби.

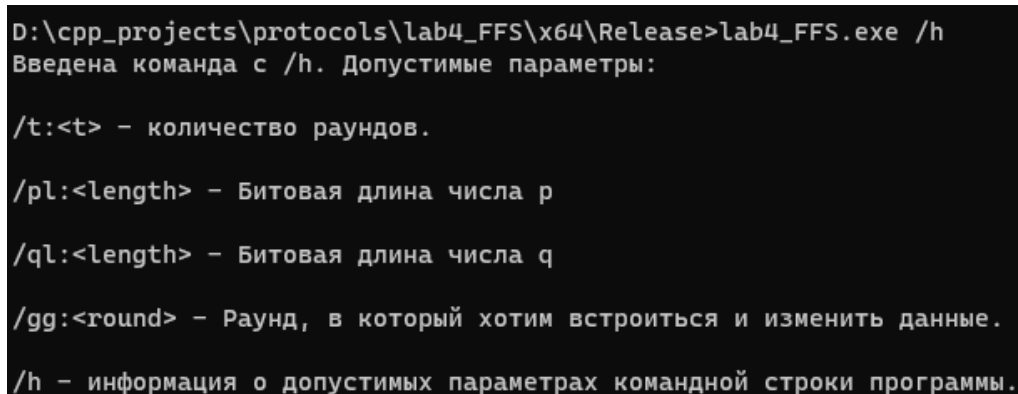
Для вычисления обратного элемента используется расширенный алгоритм Евклида, который реализован в функции *algEuclidExtended*.

Вычисление наименьшего s в кольце квадратных вычетов используется функция *findMinS*, в котором функция *findSqrt* находит квадратный корень числа по переданному модулю, а *findS*, используя китайскую теорему об остатках, находит s .

Шаги алгоритма реализованы в функции *FFS_stages*.

2.2 Тестирование программы

На рисунке 1 показан вызов параметра *help*, который выводит информацию о допустимых параметрах командной строки программы.



```
D:\cpp_projects\protocols\lab4_FFS\x64\Release>lab4_FFS.exe /h
Введена команда с /h. Допустимые параметры:

/t:<t> - количество раундов.

/pl:<length> - Битовая длина числа p

/ql:<length> - Битовая длина числа q

/gg:<round> - Раунд, в который хотим встроиться и изменить данные.

/h - информация о допустимых параметрах командной строки программы.
```

Рисунок 1 – Вызов параметра *help*

В примере, показанном на рисунках 2 и 3, задаются параметры $t = 1, p_l = 512, q_l = 512$.

```
Консольная строка
D:\cpp_projects\protocols\lab4_FFS\x64\Release>lab4_FFS.exe /t:1 /pl:512 /ql:512

Сгенерированные параметры:

p = 7433231674452364537518837965558512252577566045493012018020621657537625363854353038365436206442089085189931064128230940217853293808220266990150223705046637
q = 82281982085047019180145681668766259876641555982680899314631838364575975507526229416403901022482658820054207702623121573552403608089319360919822757176691787

n = 6116210348766340815649250050323498126569712341274250753112414384292788533701246878760110820722530668028264221844885769668791807047554774078792809981762869924580419532581
90845876122033611037869259655383597332439266083546937748822446735340599596570751588994334314618849311721474029663641826883812660209870319

Количество битов в числе n: 1023

v = 143601853117842483160225921743424486416670952988266354182238910927375679276255786221332716642632080353916648935203349885229749011746427226362746463045616025633279274107
8841678526220608237699193346171354695395871141340454466170632080102441661358597267898535837648017787890840035033853471189840721508078472740

s = 50602813113712696670680158940699726408093002615675655792193277950406881236336912076924869429705052527427230541709465558973999361003438960805953317425648604849369120197350330
1016738554430585187968799635918012495063118337513971179924097273689011382476137109025991762349056374823526577477767131636962109620465267922

РАУНД 1 ИЗ 1

[Этап 1] ПЕГГИ:
> Сгенерировала случайное число r (r < n) = 12709676386530335038884460268472759007590293609192376687783569734546864669212895062915750686195734711010038394097194221944866310
29354907091634182781603712330109195812021240760950914538458107186678699794883134615749583349743405640677949828597704280175831384853606201579366307820505292533869902130089745
009302
> Вычислила x = -(r^2) mod n = 6021849645664149855901529399830457279767258168992388291232295431746292246731731342185344740927670848255117529874665715943424847856815219277522
638941362957965415385353705783200571224292182432965274245977569352706564462990501076466200159126106005173800980093121001992191181389194449954577916801332093634159077536
> Отправила x Виктору

[Этап 2] ВИКТОР:
> Получил x от Пегги
> Выбрал случайный бит b = 0
> Отправил b Пегги
```

Рисунок 2 – Пример корректной работы упрощенной схемы идентификации Фейге-Фиата-Шамира

```
[Этап 3] ПЕГГИ:
> Получила b от Пегги
> Так как b = 0, отправила Виктору r = 1270967638653033503888446026847275900759029360919237668778356973454686466921289506291575068619573471101003839409719422194486631029354907091634182781603712330109195812021240760950914538458107186678699794883134615749583349743405640677949828597704280175831384853606201579366307820505292533869902130089745049342

[Этап 4] ВИКТОР:
> Получил от Пегги r = 1270967638653033503888446026847275900759029360919237668778356973454686466921289506291575068619573471101003839409719422194486631029354907091634182781603712330109195812021240760950914538458107186678699794883134615749583349743405640677949828597704280175831384853606201579366307820505292533869902130089745049342
> Проверил выполнение равенства x ≡ -(r^2) mod n:
x = 602184964566414985590152939983045727976725816899238829123229543174629224673173134218534474092767084825511752987466571594342484785681521927752263894136295796541538
535370578200571224292182432965274245977569352706564462990501076466200159126106005173800980093121001992191181389194449954577916801332093634159077536;
-(r^2) mod n = 6021849645664149855901529399830457279767258168992388291232295431746292246731731342185344740927670848255117529874665715943424847856815219277522638941362
957965415385353705783200571224292182432965274245977569352706564462990501076466200159126106005173800980093121001992191181389194449954577916801332093634159077536
> Так как x = -(r^2) mod n, убедился, что Пегги знает значение sqrt(x)

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОТОКОЛА В 1 РАУНДЕ: TRUE

Пройдено 1 раундов из 1

D:\cpp_projects\protocols\lab4_FFS\x64\Release>
```

Рисунок 3 – Пример корректной работы упрощенной схемы идентификации Фейге-Фиата-Шамира

В примере, показанном на рисунках 4, 5 и 6, изменим на девятом раунде x на другое число.

```

D:\cpp_projects\protocols\lab4_FFS\x64\Release>lab4_FFS.exe /t:10 /pl:32 /ql:32 /gg:9

Сгенерированные параметры:

p = 2757167239

q = 2972508947

n = 8195704286302787333

Количество битов в числе n: 63

v = 3228520870764946611

s = 5088166133603834357

РАУНД 1 ИЗ 10

```

Рисунок 4 – Пример работы упрощенной схемы идентификации Фейге-Фиата-Шамира, где встраиваются и меняют значение на 9 раунде

```

Командная строка
РАУНД 9 ИЗ 10

[Этап 1] ПЕГГИ:
> Сгенерировала случайное число r (r < n) = 7797329440104147614
> Вычислила x = -(r^2) mod n = 3838317236178553633
> Отправила x Виктору

КОНЕЦ ЭТАПА 1. ПРОДОЛЖИТЬ - y, ИЗМЕНИТЬ - i : i

x = 3838317236178553633

Можно изменить: 1 - x
Ваш выбор: 1

Введите новое значение: 13221421

[Этап 2] ВИКТОР:
> Получил x от Пегги
> Выбрал случайный бит b = 0
> Отправил b Пегги

КОНЕЦ ЭТАПА 2. ПРОДОЛЖИТЬ - y, ИЗМЕНИТЬ - i : y

[Этап 3] ПЕГГИ:
> Получила b от Пегги
> Так как b = 0, отправила Виктору r = 7797329440104147614

КОНЕЦ ЭТАПА 3. ПРОДОЛЖИТЬ - y, ИЗМЕНИТЬ - i : y

[Этап 4] ВИКТОР:
> Получил от Пегги r = 7797329440104147614
> Проверил выполнение равенства x = -(r^2) mod n:
  x = 13221421;
  -(r^2) mod n = 3838317236178553633
> Так как x != -(r^2) mod n, убедился, что Пегги НЕ знает значение sqrt(x)

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОТОКОЛА В 9 РАУНДЕ: FALSE

```

Рисунок 5 – Пример работы упрощенной схемы идентификации Фейге-Фиата-Шамира, где встраиваются и меняют значение на 9 раунде


```
РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОТОКОЛА В 10 РАУНДЕ: TRUE
```

```
Пройдено 9 раундов из 10
```

```
D:\cpp_projects\protocols\lab4_FFS\x64\Release>
```

Рисунок 6 – Пример работы упрощенной схемы идентификации Фейге-Фиата-Шамира, где встраиваются и меняют значение на 9 раунде

ПРИЛОЖЕНИЕ А

Листинг программы

```
#include <iostream>
#include <vector>
#include <chrono>
#include <time.h>
#include <boost/random/random_device.hpp>
#include <boost/multiprecision/cpp_int.hpp>
#include <boost/random.hpp>
#include <sstream>
#include <fstream>
#include <unordered_map>
#include <string>
#include <windows.h>
#include <cryptlib.h>
#include "rijndael.h"
#include "modes.h"
#include "files.h"
#include "osrng.h"
#include "hex.h"
#include <unordered_set>

using namespace std;
using namespace boost::multiprecision;
using namespace boost::random;
using namespace CryptoPP;

const int AES_KEY_SIZE = AES::DEFAULT_KEYLENGTH;
const int AES_BLOCK_SIZE = AES::BLOCKSIZE;
cpp_int pSize;

cpp_int rand_large(cpp_int w1, cpp_int w2) {
    random_device gen;
    uniform_int_distribution<cpp_int> ui(w1, w2);
    cpp_int y = ui(gen);
    return y;
}

cpp_int generate_random(cpp_int a, cpp_int b) {
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<cpp_int> dist(a, b);
    return dist(gen);
}

cpp_int rand_large_by_bit_length(int l) {
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<int> distribution(0, 1);
    cpp_int result = 0;
    for (int i = 1; i < l - 1; ++i) {
        result <<= 1;
        result += distribution(gen);
    }
    result |= (cpp_int(1) << (l - 1));
    result |= 1;
    return result;
}
```

```

cpp_int powMod(cpp_int x, cpp_int n, cpp_int m) {
    cpp_int N = n, Y = 1, Z = x % m;
    while (N != 0) {
        cpp_int lastN = N % 2;
        N = N / 2;
        if (lastN == 0) {
            Z = (Z * Z) % m;
            continue;
        }
        Y = (Y * Z) % m;
        if (N == 0)
            break;
        Z = (Z * Z) % m;
    }
    return Y % m;
}

cpp_int nod(cpp_int a, cpp_int m) {
    if (m == 0)
        return a;
    else
        return nod(m, a % m);
}

cpp_int algEuclidExtended(cpp_int a, cpp_int b, cpp_int& x, cpp_int& y) {
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    cpp_int xi, yi;
    cpp_int nod = algEuclidExtended(b % a, a, xi, yi);
    x = yi - (b / a) * xi;
    y = xi;
    return nod;
}

bool isPrimeMillerRabin(cpp_int n, cpp_int k) {
    if (n <= 1 || n == 4) return false;
    if (n <= 3) return true;

    cpp_int d = n - 1;
    while (d % 2 == 0)
        d /= 2;

    for (cpp_int i = 0; i < k; i++) {
        cpp_int a = 2 + rand() % (n - 3);
        cpp_int x = powMod(a, d, n);

        if (x == 1 || x == n - 1)
            continue;

        while (d != n - 1) {
            x = (x * x) % n;
            d *= 2;

            if (x == 1) return false;
            if (x == n - 1) break;
        }

        if (x != n - 1) return false;
    }

    return true;
}

```

```

}

string findInStr(string const& str, int n) {
    if (str.length() < n)
        return str;
    return str.substr(0, n);
}

cpp_int min_num_by_l_bits(int l) {
    cpp_int result = 1;
    result |= (cpp_int(1) << (l - 1));
    return result;
}

cpp_int max_num_by_l_bits(int l) {
    cpp_int result = (cpp_int(1) << l) - 1;
    return result;
}

cpp_int generateRandomPrime(int l) {
    cpp_int minNum = min_num_by_l_bits(l);
    cpp_int maxNum = max_num_by_l_bits(l);
    cpp_int randNum = rand_large_by_bit_length(l);
    cpp_int newNum = randNum;
    bool plus1 = true;
    if (newNum % 2 == 0) {
        if (newNum + 1 < maxNum)
            newNum += 1;
        else {
            newNum -= 1;
            plus1 = false;
        }
    }
    randNum = newNum;
    while (!isPrimeMillerRabin(newNum, 20)) {
        if (plus1)
            newNum += 2;
        else
            newNum -= 2;
        if (newNum > maxNum) {
            newNum = randNum;
            plus1 = false;
        }
        if (newNum < minNum) {
            newNum = randNum;
            plus1 = true;
        }
    }
    return newNum;
}

vector<string> splitString(const string& input, char zn) {
    istringstream stream(input);
    string str1;
    vector<string> strs;
    while (getline(stream, str1, zn)) {
        strs.push_back(str1);
    }
    return strs;
}

void generate_n(int pLength, int qLength, cpp_int &p, cpp_int &q, cpp_int &n) {
    p = generateRandomPrime(pLength);
    while (!isPrimeMillerRabin(p, 20)) {

```

```

        p = generateRandomPrime(pLength);
    }
    q = generateRandomPrime(qLength);
    while (!isPrimeMillerRabin(q, 20)) {
        q = generateRandomPrime(qLength);
    }
    n = p * q;
    return;
}

cpp_int jacobi_symbol(cpp_int a, cpp_int b) {
    if (nod(a, b) != 1)
        return 0;
    else {
        cpp_int r = 1;
        while (a != 0) {
            cpp_int t = 0;
            while (a % 2 == 0) {
                t = t + 1;
                a = a / 2;
            }
            if (t % 2 != 0)
                if (powMod(b, 1, 8) == 3 || powMod(b, 1, 8) == 5)
                    r = r * (-1);
            if (powMod(a, 1, 4) == 3 && powMod(b, 1, 4) == 3)
                r = r * (-1);
            cpp_int c = a;
            if (c != 0)
                a = powMod(b, 1, c);
            b = c;
        }
        return r;
    }
}

void generatePublicKey(cpp_int n, cpp_int p, cpp_int q, cpp_int &peggyPublicKey) {
    while (true) {
        cpp_int v = generate_random(2, n - 2);
        if (nod(v, n) != 1 || jacobi_symbol(v, p) != 1 || jacobi_symbol(v, q) != 1)
            continue;
        cpp_int vSqr = v * v;
        cpp_int phi = (p - 1) * (q - 1);
        cpp_int deg = phi / 2;
        if (powMod(v, deg, n) == 1) {
            peggyPublicKey = v;
            return;
        }
    }
    return;
}

cpp_int normalMod(cpp_int x, cpp_int p) {
    x %= p;
    while (x < 0)
        x += p;
    return x;
}

bool findQM(const cpp_int& p, cpp_int& q, cpp_int& m) {
    if (p <= 1 || p % 2 == 0)
        return false;
    cpp_int k = p - 1;
    m = 0;
    while (k % 2 == 0) {

```

```

        k /= 2;
        m++;
    }
    q = k;
    if (nod(q, 2) == 1)
        return true;
    else
        return false;
}

void helpFunc() {
    cout << "Введена команда с /h. Допустимые параметры:";
    cout << "\n\n/t:<t> - количество раундов.";
    cout << "\n\n/pl:<length> - Битовая длина числа p";
    cout << "\n\n/ql:<length> - Битовая длина числа q";
    cout << "\n\n/gg:<round> - Раунд, в который хотим встроиться и изменить данные.";
    cout << "\n\n/h - информация о допустимых параметрах командной строки программы.\n";
}

cpp_int pow2(cpp_int s, cpp_int k) {
    if (k == 0)
        return 1;
    else if (k == 1)
        return s;
    cpp_int s_start = s;
    for (int i = 0; i < k - 1; i++)
        s *= s_start;
    return s;
}

cpp_int func_k_i(cpp_int a_i, cpp_int q, cpp_int p) {
    cpp_int k = 0;
    while (powMod(a_i, pow2(2, k) * q, p) != 1)
        k++;
    return k;
}

cpp_int x_0(cpp_int q, cpp_int p, cpp_int m, cpp_int b, cpp_int a) {
    vector <cpp_int> aVec;
    vector <cpp_int> kVec;
    vector <cpp_int> rVec;
    aVec.push_back(a);
    cpp_int k_i = func_k_i(aVec[0], q, p);
    kVec.push_back(k_i);
    int a_index = 0;
    while (true) {
        cpp_int a_i = aVec[a_index];
        k_i = func_k_i(a_i, q, p);
        if (k_i == 0)
            break;
        if (a_index != 0)
            kVec.push_back(k_i);
        cpp_int a_iB = (a_i * pow2(b, pow2(2, m - k_i))) % p;
        aVec.push_back(a_iB);
        a_index++;
    }
    int n = aVec.size();
    rVec.resize(n);
    rVec[n - 1] = powMod(aVec[n - 1], (q + 1) / 2, p);
    for (int r_i = n - 2; r_i >= 0; r_i--) {
        cpp_int x1, y1;
        cpp_int b_deg = pow2(b, pow2(2, m - kVec[r_i] - 1)) % p; //p
        algEuclidExtended(b_deg, p, x1, y1);
        if (x1 < 0)

```

```

        x1 = x1 + p;
        cpp_int el_obr = x1;
        rVec[r_i] = (rVec[r_i + 1] * el_obr) % p;
    }

    return rVec[0];
}

cpp_int findSqrt(cpp_int p, cpp_int v) {
    cpp_int q, m;
    if (!findQM(p, q, m)) {
        cout << "p - 1 не раскладывается как 2^m * q\n";
        return -1;
    }
    cpp_int b = generate_random(2, p - 1);
    while (jacobi_symbol(b, p) != -1) {
        b = generate_random(2, p - 1);
    }
    cpp_int x0 = x_0(q, p, m, b, v);

    return x0;
}

cpp_int findS(cpp_int x1, cpp_int x2, cpp_int p, cpp_int q, cpp_int n) {
    cpp_int s1 = normalMod(x1, p);
    cpp_int s2 = normalMod(x2, q);
    cpp_int k1, k2;
    algEuclidExtended(p, q, k1, k2);
    if (k1 < 0)
        k1 = k1 + q;
    if (k2 < 0)
        k2 = k2 + p;

    // Используем Китайскую теорему об остатках для нахождения s.
    cpp_int s = (s1 * q * k2 + s2 * p * k1) % n;
    return s;
}

cpp_int findMinS(cpp_int p, cpp_int q, cpp_int n, cpp_int v, cpp_int v_obr) {
    cpp_int v1 = normalMod(v_obr, p);
    cpp_int x1 = findSqrt(p, v1);
    cpp_int v2 = normalMod(v_obr, q);
    cpp_int x2 = findSqrt(q, v2);
    cpp_int s = findS(x1, x2, p, q, n);
    return s;
}

bool FFS_stages(cpp_int p, cpp_int q, cpp_int n, cpp_int peggyPublicKey, cpp_int
peggyPrivateKey, bool hack) {

    //ШАГ 1

    cout << "\n\n[Этап 1] ПЕГГИ:\n";

    cpp_int Peggy_r = generate_random(2, n - 1);
    cout << "\n > Сгенерировала случайное число r (r < n) = " << Peggy_r;
    cpp_int r2 = normalMod(normalMod(-Peggy_r, n) * normalMod(-Peggy_r, n), n);
    cpp_int Peggy_x = normalMod(r2, n);
    cout << "\n > Вычислила x = -(r^2) mod n = " << Peggy_x;
    cout << "\n > Отправила x Виктору";

    cpp_int Peggy_x_IzM = Peggy_x;
    if (hack) {

```

```

string userCout;
cout << "\n\nКОНЕЦ ЭТАПА 1. ПРОДОЛЖИТЬ - y, ИЗМЕНИТЬ - i : ";
cin >> userCout;
if (userCout == "i") {
    cout << "\n\nx = " << Peggy_x;
    cout << "\n\nМожно изменить: 1 - x";
    cout << "\nВаш выбор: ";
    string userChoiceIzm;
    cin >> userChoiceIzm;
    if (userChoiceIzm == "1") {
        cout << "\n\nВведите новое значение: ";
        cin >> Peggy_x_IzM;
    }
}
else if (userCout != "y")
    return false;
}

//ШАГ 2

cout << "\n\n[Этап 2] ВИКТОР:\n";

cpp_int Victor_x = Peggy_x_IzM;
cout << "\n > Получил x от Пегги";
cpp_int Victor_b = generate_random(0, 1);
cout << "\n > Выбрал случайный бит b = " << Victor_b;
cout << "\n > Отправил b Пегги";

cpp_int Victor_b_IzM = Victor_b;
if (hack) {
    string userCout;
    cout << "\n\nКОНЕЦ ЭТАПА 2. ПРОДОЛЖИТЬ - y, ИЗМЕНИТЬ - i : ";
    cin >> userCout;
    if (userCout == "i") {
        cout << "\n\nx = " << Peggy_x;
        cout << "\n\nМожно изменить: 1 - b";
        cout << "\nВаш выбор: ";
        string userChoiceIzm;
        cin >> userChoiceIzm;
        if (userChoiceIzm == "1") {
            cout << "\n\nВведите новое значение: ";
            cin >> Victor_b_IzM;
        }
    }
    else if (userCout != "y")
        return false;
}

//ШАГ 3

cout << "\n\n[Этап 3] ПЕГГИ:\n";

cpp_int Peggy_b = Victor_b_IzM;
cout << "\n > Получила b от Пегги";
cpp_int forVictor;
if (Peggy_b == 0) {
    forVictor = Peggy_r;
    cout << "\n > Так как b = 0, отправила Виктору r = " << forVictor;
}
else if (Peggy_b == 1) {
    forVictor = normalMod(Peggy_r * peggyPrivateKey, n);
    cout << "\n > Так как b = 1, отправила Виктору y = r * s (mod n) = " << forVictor;
}
else {

```



```

        cout << "\n > ERROR: b != 0 и b != 1 ";
        return false;
    }

    cpp_int forVictor_Izm = forVictor;
    if (hack) {
        string userCout;
        cout << "\n\nКОНЕЦ ЭТАПА 3. ПРОДОЛЖИТЬ - y, ИЗМЕНИТЬ - i : ";
        cin >> userCout;
        if (userCout == "i") {
            cout << "\n\ny or r = " << forVictor;
            cout << "\n\nМожно изменить: 1 - y or r";
            cout << "\nВаш выбор: ";
            string userChoiceIzm;
            cin >> userChoiceIzm;
            if (userChoiceIzm == "1") {
                cout << "\n\nВведите новое значение: ";
                cin >> forVictor_Izm;
            }
        }
        else if (userCout != "y")
            return false;
    }

    //ШАГ 4

    cout << "\n\n[Этап 4] ВИКТОР:\n";
    cpp_int Victor_4 = forVictor_Izm;
    if (Victor_b == 0) {
        cout << "\n > Получил от Пегги r = " << Victor_4;
        cpp_int rVic = normalMod(normalMod(normalMod(-Victor_4, n) * normalMod(-Victor_4,
n), n), n);
        cout << "\n > Проверил выполнение равенства  $x = -(r^2) \bmod n$ : \n\tx = " << Victor_x
<< "; \n\t-(r^2) \bmod n = " << rVic;
        if (Victor_x == rVic) {
            cout << "\n > Так как  $x = -(r^2) \bmod n$ , убедился, что Пегги знает значение
sqrt(x)";
            return true;
        }
        else {
            cout << "\n > Так как  $x \neq -(r^2) \bmod n$ , убедился, что Пегги НЕ знает значение
sqrt(x)";
            return false;
        }
    }
    else if (Victor_b == 1) {
        cout << "\n > Получил от Пегги  $y = r * s \pmod n$  = " << Victor_4;
        cpp_int rVic = normalMod(normalMod(normalMod(Victor_4 * Victor_4, n) *
peggyPublicKey, n), n);
        cout << "\n > Проверил выполнение равенства  $x = y^2 * v \pmod n$ : \n\tx = " <<
Victor_x << "; \n\ty^2 * v \pmod n = " << rVic;
        if (Victor_x == rVic) {
            cout << "\n > Так как  $x = y^2 * v \pmod n$ , убедился, что Пегги знает значение
sqrt(v^(-1))";
            return true;
        }
        else {
            cout << "\n > Так как  $x \neq y^2 * v \pmod n$ , убедился, что Пегги НЕ знает
значение sqrt(v^(-1))";
            return false;
        }
    }
    else {
        cout << "\n > ERROR: b != 0 и b != 1 ";
    }
}

```

```

        return false;
    }
}

bool FFS_main(int pLength, int qLength, cpp_int t, cpp_int ggRound) {
    cpp_int p, q, n;
    generate_n(pLength, qLength, p, q, n);
    cpp_int peggyPublicKey;
    generatePublicKey(n, p, q, peggyPublicKey);

    cpp_int x1, y1;
    algEuclidExtended(peggyPublicKey, n, x1, y1);
    if (x1 < 0)
        x1 = x1 + n;
    cpp_int v_obr = x1;
    cpp_int peggyPrivateKey = findMinS(p, q, n, peggyPublicKey, v_obr);
    cout << "\nСгенерированные параметры:\n";
    cout << "\np = " << p << "\n";
    cout << "\nq = " << q << "\n";
    cout << "\nn = " << n << "\n";
    int bit_count = msb(n) + 1;
    cout << "\nКоличество битов в числе n: " << bit_count << "\n";
    cout << "\nv = " << peggyPublicKey << "\n";
    cout << "\ns = " << peggyPrivateKey << "\n";
    cpp_int rounds = 1;
    cpp_int true_rounds = 0;
    while (rounds != t + 1) {
        bool hack = false;
        cout << "\n\n\tТАУНД " << rounds << " ИЗ " << t << "\n\n";
        if (rounds == ggRound)
            hack = true;
        if (FFS_stages(p, q, n, peggyPublicKey, peggyPrivateKey, hack)) {
            cout << "\nРЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОТОКОЛА В " << rounds << " РАУНДЕ: TRUE\n";
            true_rounds++;
        }
        else
            cout << "\nРЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОТОКОЛА В " << rounds << " РАУНДЕ: FALSE\n";
        rounds++;
    }
    cout << "\n\nПройдено " << true_rounds << " раундов из " << t << "\n";
    return true;
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");
    int lenP = 0, lenQ = 0, t = 0;
    cpp_int ggRound = 0;
    for (int i = 0; argv[i]; i++) {
        string checkStr = string(argv[i]);
        if (findInStr(checkStr, 2) == "/h") {
            helpFunc();
            return 0;
        }
        if (checkStr.length() > 2) {
            string ifStr = findInStr(checkStr, 3);
            char symbol = ',';
            if (ifStr == "/t:") {
                t = stoi(checkStr.substr(3, checkStr.length()));
            }
            if (ifStr == "/p1") {
                lenP = stoi(checkStr.substr(4, checkStr.length()));
            }
            if (ifStr == "/q1") {

```

```

        lenQ = stoi(checkStr.substr(4, checkStr.length()));
    }
    if (ifStr == "/gg") {
        ggRound = stoi(checkStr.substr(4, checkStr.length()));
    }
}

FFS_main(lenP, lenQ, t, ggRound);

return 0;
}

```