

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

**Упрощённая схема идентификации Feige-Fiat-Shamir**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Алексеева Александра Александровича

Преподаватель

профессор, д.ф.-м.н.

\_\_\_\_\_

В. Е. Новиков

подпись, дата

Саратов 2023

## СОДЕРЖАНИЕ

1 Теоретическая часть.....	3
1.1 Описание алгоритма.....	3
2 Описание программы.....	5
2.1 Пример работы программы.....	5
3 Листинг кода.....	6

## 1 Теоретическая часть

Цель работы – изучение упрощённой схемы идентификации Фейге-Фиата-Шамира.

### 1.1 Описание алгоритма

Перед выдачей любых закрытых ключей арбитр выбирает случайный модуль,  $n$ , который является произведением двух больших простых чисел. В реальной жизни длина  $n$  должна быть не меньше 512 битов и лучше как можно ближе к 1024 битам.  $n$  может общим для группы контролеров (Использование чисел Блума (Blum) облегчит вычисления, но не является обязательным для безопасности).

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

(1) Пегги выбирает случайное  $r$ , меньшее  $n$ . Затем она вычисляет  $x = -r^2 \pmod{n}$  и посылает  $x$  Виктору.

(2) Виктор посылает Пегги случайный бит  $b$ .

(3) Если  $b = 0$ , то Пегги посылает Виктору  $r$ . Если  $b = 1$ , то Пегги посылает Виктору  $y = r \cdot s \pmod{n}$ .

(4) Если  $b = 0$ , Виктор проверяет, что  $x = -r^2 \pmod{n}$ , убеждаясь, что Пегги знает значение  $\text{sqrt}(x)$ . Если  $b = 1$ , Виктор проверяет, что  $x = y^2 \cdot v \pmod{n}$ , убеждаясь, что Пегги знает значение  $\text{sqrt}(v^{-1})$ .

Это один этап протокола, называемый **аккредитацией**. Пегги и Виктор повторяют этот протокол  $t$  раз, пока Виктор не убедится, что Пегги знает  $s$ . Это протокол "разрезать и выбрать". 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  $1/2^t$ .

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  $1/2$ . Следовательно, вероятность, что он обманет Валерию, равна 50 процентам. Вероятность, что ему удастся обмануть ее  $t$  раз, равна  $1/2^t$ .

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 всё закончится.

## 2 Описание программы

Программа, представленная ниже, содержит следующие функции:

- `powMy(x, y)` – возводит число  $x$  в степень  $y$ ;
- `powClosed(x, y, mod)` – возводит число  $x$  в степень  $y$  по модулю  $mod$ ;
- `usualEuclid(a, b)` – вычисление НОД чисел  $a$  и  $b$  обычным алгоритмом Евклида;
- `advancedEuclid(a, b)` – вычисляет НОД( $a, b$ ) и представляет его в виде  $ax + by$ ;
- `symbolLegendre(a, p)` – вычисление символа Лежандра  $\left(\frac{a}{p}\right)$ ;
- `miller_rabin(n, k = 10)` – проверка числа  $n$  на простоту с вероятностью  $\frac{1}{2^k}$ ;
- `generatesimpleNum()` – генерация простого числа размером 32-64 бит;
- `sqrtFromZp(a, p)` – извлечение квадратного корня из  $a$  в поле  $p$ ;
- `minS(p, q, n, v)` – извлечение квадратного корня из  $v$  по модулю  $n = p \cdot q$ , где  $p$  и  $q$  – большие простые числа;
- `feigeFiatShamir()` – упрощённая схема идентификации Фейге-Фиата-Шамира.

### 2.1 Примеры работы программы

```
Упрощённая схема идентификации Фейге-Фиата-Шамира
Доверенный центр выбирает случайный модуль n = 416919215094865416326883526312988477

Открытый ключ Пегги v = 2331091105300953075548160000
Закрытый ключ Пегги s = 301495433463843935184548201289899084

Пегги выбирает случайное r = 727433282, вычисляет x = -r^2 (mod n) = 529159179761291524 и отправляет x Виктору
Виктор посылает Пегги случайный бит b = 0
Т.к. бит b = 0, то Пегги посылает Виктору r = 727433282
Т.к. бит b = 0, то Виктор проверяет, что x = -r^2 (mod n) = 529159179761291524, убеждаясь, что Пегги знает значение sqrt(x)

Повторить протокол y/n? y

Пегги выбирает случайное r = 1366766368, вычисляет x = -r^2 (mod n) = 1868050304695911424 и отправляет x Виктору
Виктор посылает Пегги случайный бит b = 1
Т.к. бит b = 1, то Пегги посылает Виктору y = r * s (mod n) = 230362955425061566754464940409961028
Т.к. бит b = 1, то Виктор проверяет, что x = y^2 * v (mod n) = 1868050304695911424, убеждаясь, что Пегги знает значение sqrt(v^-1)

Повторить протокол y/n?
```

### 3 Листинг кода

```
#include "iostream"
#include "cmath"
#include "vector"
#include "boost/multiprecision/cpp_int.hpp"

using namespace std;
using namespace boost::multiprecision;

cpp_int powMy(cpp_int x, cpp_int y) {
    cpp_int res = 1;
    for (int i = 0; i < y; i++)
        res *= x;
    return res;
}

vector <cpp_int> deg2(cpp_int el, cpp_int n) { //Раскладываем число на степени
двойки
    vector <cpp_int> res;
    while (n != 0) {
        if (n / el == 1) {
            res.push_back(el);
            n -= el;
            el = 1;
        }
        else
            el *= 2;
    }
    return res;
}

cpp_int multMod(cpp_int n, cpp_int mod, vector <pair <cpp_int, cpp_int>> lst)
{ //Умножаем число по модулю
    if (lst.size() == 1) {
        cpp_int res = 1;
        for (int i = 0; i < lst[0].second; i++)
            res = res * lst[0].first % mod;
        return res;
    }
    else if (lst[0].second == 1) {
        cpp_int el = lst[0].first;
        lst.erase(lst.begin());
        return (el * multMod(n, mod, lst)) % mod;
    }
    else {
        for (int i = 0; i < lst.size(); i++)
            if (lst[i].second > 1) {
                lst[i].first = (lst[i].first * lst[i].first) %
mod;
                lst[i].second /= 2;
            }
        return multMod(n, mod, lst);
    }
}

cpp_int powClosed(cpp_int x, cpp_int y, cpp_int mod) { //Возводим число в степени
по модулю
    if (y == 0)
        return 1;
}
```

```

vector <cpp_int> lst = deg2(1, y);
vector <pair <cpp_int, cpp_int>> xDeps;
for (int i = 0; i < lst.size(); i++)
    xDeps.push_back(make_pair(x, lst[i]));

cpp_int res = multMod(x, mod, xDeps);
return res;
}

cpp_int decForm(string x) {
    cpp_int res = 0, deg = 1;
    if (x.back() == '1')
        res += 1;
    for (int i = 1; i < x.length(); i++) {
        deg = deg * 2;
        if (x[x.length() - i - 1] == '1')
            res += deg;
    }
    return res;
}

cpp_int usualEuclid(cpp_int a, cpp_int b) {
    if (a < b)
        swap(a, b);
    if (a < 0 || b < 0)
        throw string{ "Выполнение невозможно: а < 0 или b < 0" };
    else if (b == 0)
        return a;

    cpp_int r = a % b;
    return usualEuclid(b, r);
}

pair <cpp_int, cpp_int> advancedEuclid(cpp_int a, cpp_int b) {
    if (a < 0 || b < 0)
        throw string{ "Выполнение невозможно: а < 0 или b < 0" };

    cpp_int q, aPrev = a, aCur = b, aNext = -1;
    cpp_int xPrev = 1, xCur = 0, xNext;
    cpp_int yPrev = 0, yCur = 1, yNext;
    while (aNext != 0) {
        q = aPrev / aCur;
        aNext = aPrev % aCur;
        aPrev = aCur; aCur = aNext;

        xNext = xPrev - (xCur * q);
        xPrev = xCur; xCur = xNext;

        yNext = yPrev - (yCur * q);
        yPrev = yCur; yCur = yNext;
    }

    return make_pair(xPrev, yPrev);
}

cpp_int funEuler(cpp_int n) {
    cpp_int res = 1;
    for (int i = 2; i < n; i++)

```

```

        if (usualEuclid(n, i) == 1)
            res++;
    return res;
}

bool miller_rabin(cpp_int n, int k = 10) {
    if (n == 0 || n == 1)
        return false;

    cpp_int d = n - 1;
    cpp_int s = 0;
    while (d % 2 == 0) {
        s++;
        d = d / 2;
    }

    cpp_int nDec = n - 1;
    for (int i = 0; i < k; i++) {
        cpp_int a = rand() % nDec;
        if (a == 0 || a == 1)
            a = a + 2;

        cpp_int x = powClosed(a, d, n);
        if (x == 1 || x == nDec)
            continue;

        bool flag = false;
        for (int j = 0; j < s; j++) {
            x = (x * x) % n;
            if (x == nDec) {
                flag = true;
                break;
            }
        }
        if (!flag)
            return false;
    }

    return true;
}

cpp_int symbolLegendre(cpp_int a, cpp_int p) {
    if (a == 0)
        return 0;
    cpp_int res = powClosed(a, (p - 1) / 2, p);
    return res == 1 ? 1 : -1;
}

cpp_int generateSimpleNum() {
    cpp_int q = rand() % 1000;
    while (funEuler(q) != q - 1)
        q++;

    cpp_int s, n = 2, nDec;
    while (!miller_rabin(n)) {
        string sBin = "";
        int sBinSize = 32 + rand() % 32;
        for (int i = 0; i < sBinSize; i++)
            sBin = sBin + to_string(rand() % 2);
        s = decForm(sBin);
    }
}

```



```

        n = (q * s) + 1;
        nDec = n - 1;
    }

    return n;
}

cpp_int sqrtFromZp(cpp_int a, cpp_int p) {
    a = a % p;
    cpp_int m = 0, q = p - 1;
    while (q % 2 != 1) {
        m++;
        q /= 2;
    }

    cpp_int b = rand() % p;
    while (symbolLegendre(b, p) != -1)
        b = (b + 1) % p;

    vector<cpp_int> kArr;
    for (int i = 1;; i++) {
        cpp_int k = 0;
        while (powClosed(a, powMy(2, k) * q, p) != 1)
            k++;
        kArr.push_back(k);
        if (k == 0)
            break;
        a = (a * powMy(b, powMy(2, m - kArr.back()))) % p;
    }

    cpp_int r = powClosed(a, (q + 1) / 2, p);
    for (int i = kArr.size() - 2; i >= 0; i--)
        r = (r * advancedEuclid(powMy(b, powMy(2, m - kArr[i] - 1)),
p).first) % p;

    return r;
}

cpp_int minS(cpp_int p, cpp_int q, cpp_int n, cpp_int vRev) {
    cpp_int s1 = sqrtFromZp(vRev, p), s2 = sqrtFromZp(vRev, q);

    pair<cpp_int, cpp_int> res = advancedEuclid(p, q);
    if (res.first < 0)
        res.first += q;
    if (res.second < 0)
        res.second += p;

    return (s1 * q * res.second + s2 * p * res.first) % n;
}

void feigeFiatShamir() {
    cpp_int p = generateSimpleNum(), q = generateSimpleNum();
    cpp_int n = p * q;
    cout << "\nДоверенный центр выбирает случайный модуль n = " << n;

    cpp_int x = powMy(abs(rand() * rand()), 2) % n;
    cpp_int v = x * x % n;
    cpp_int vRev = advancedEuclid(v, n).first;
    while (vRev < 0)

```

```

        vRev += n;
        cout << "\n\nОткрытый ключ Пегги v = " << v;

        cpp_int s = minS(p, q, n, vRev);
        if (s < 0)
            s += n;
        cout << "\n\nЗакрытый ключ Пегги s = " << s;

        for (;;) {
            cpp_int r = abs(rand() * rand() * rand()) % n;
            cout << "\n\nПегги выбирает случайное r = " << r;
            cpp_int x = (-r + n) * (-r + n) % n;
            cout << ", вычисляет  $x = -r^2 \pmod n$  = " << x << " и отправляет
x Виктору";

            short b = rand() % 2;
            cout << "\n\nВиктор посылает Пегги случайный бит b = " << b;

            if (b == 0) {
                cout << "\nТ.к. бит b = 0, то Пегги посылает Виктору r =
" << r;
                cout << "\nТ.к. бит b = 0, то Виктор проверяет, что  $x = -r^2 \pmod n$  = " << (-r + n) * (-r + n) % n << ", убеждаясь, что Пегги знает
значение  $\sqrt{x}$ ";
            }
            else {
                cpp_int y = r * s % n;
                cout << "\nТ.к. бит b = 1, то Пегги посылает Виктору y =
r * s  $\pmod n$  = " << y;
                cout << "\nТ.к. бит b = 1, то Виктор проверяет, что  $x = y^2 * v \pmod n$  = " << y * y * v % n << ", убеждаясь, что Пегги знает значение
 $\sqrt{v^{-1}}$ ";
            }

            takeChoice:
            cout << "\n\nПовторить протокол y/n? ";
            char choice;
            cin >> choice;
            if (choice == 'y')
                continue;
            else if (choice == 'n')
                return;
            else {
                cout << "Incorrect! Try again";
                goto takeChoice;
            }
        }
    }

int main() {
    srand(time(0));
    setlocale(LC_ALL, "ru");
    cout << "\tУпрощённая схема идентификации Фейге-Фиата-Шамира";

    feigeFiatShamir();
    cout << endl;
    return 0;
}

```