

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

Базовый протокол (алгоритм Диффи-Хеллмана)

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

аспирант

Р. А. Фарахутдинов

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.....	7

1 Теоретическая часть

Цель работы – изучение базового протокола (алгоритма Диффи-Хеллмана).

1.1 Описание алгоритма

Diffie-Hellman, первый в истории алгоритм с открытым ключом, был изобретен в 1976 году. Его безопасность опирается на трудность вычисления дискретных логарифмов в конечном поле (в сравнении с легкостью возведения в степень в том же самом поле). Diffie-Hellman может быть использован для распределения ключей – Алиса и Боб могут воспользоваться этим алгоритмом для генерации секретного ключа – но его нельзя использовать для шифрования и дешифрирования сообщений.

Математика несложна. Сначала Алиса и Боб вместе выбирают большие простые числа n и g так, чтобы g было примитивом $\text{mod } n$. Эти два целых числа хранить в секрете необязательно, Алиса и Боб могут договориться об их использовании по несекретному каналу. Эти числа даже могут совместно использоваться группой пользователей. Без разницы. Затем выполняется следующий протокол:

(1) Алиса выбирает случайное большое целое число x и посылает Бобу
$$X = g^x \text{ mod } n$$

(2) Боб выбирает случайное большое целое число y и посылает Алисе
$$Y = g^y \text{ mod } n$$

(3) Алиса вычисляет
$$k = Y^x \text{ mod } n$$

(4) Боб вычисляет
$$k' = X^y \text{ mod } n$$

И k , и k' равны $g^{xy} \text{ mod } n$. Никто из подслушивающих этот канал не сможет вычислить это значение, им известно только n , g , X и Y . Пока они не смогут вычислить дискретный логарифм и раскрыть x или y , они не смогут решить

проблему. Поэтому, k – это секретный ключ, который Алиса и Боб вычисляют независимо.

Выбор g и n может заметно влиять на безопасность системы. Число $(n-1)/2$ также должно быть простым. И, самое главное, n должно быть большим: безопасность системы основана на сложности разложения на множители чисел того же размера, что и n . Можно выбирать любое g , которое является примитивом $\text{mod } n$; нет причин, по которым нельзя было бы выбрать наименьшее возможное g – обычно одноразрядное число. (К тому же, на самом деле, g не должно даже быть примитивом, оно только должно генерировать достаточно большую подгруппу мультипликативной группы $\text{mod } n$).

2 Описание программы

Программа, представленная ниже, содержит следующие функции:

- `powClosed (x, y, mod)` – возводит число x в степень y по модулю mod ;
- `miller_rabin (n, k = 10)` – проверка числа n на простоту с вероятностью $\frac{1}{2^k}$;
- `generateGN()` – генерация чисел g и n : g является первообразным корнем для n ;
- `diffHell2(g, n, keys)` – генерация общего секретного ключа $g^{keys[0] \cdot keys[1]} \pmod n$ для двух пользователей;
- `diffHell(g, n, keys)` – генерация общего секретного ключа $g^{keys[0] \cdot keys[1] \cdot \dots \cdot keys[k-1]} \pmod n$ для k пользователей;

2.1 Примеры работы программы

```
Базовый протокол (алгоритм Диффи-Хеллмана)
Количество пользователей: 2

Выбрать открытые параметры g и n вручную y/n? n
Открытый параметр g = 45187474728371910
Открытый параметр n = 56201615489360947

Пользователи будут выбирать секретные большие числа вручную u/n? n
1-й пользователь выбирает число a = 2250504
2-й пользователь выбирает число b = 47838431

1-й пользователь вычисляет ключ key = 55196239687216262 и передаёт 2-му пользователю. 2-й пользователь вычисляет
key = 54454104531645922 и получает общий секретный ключ.
2-й пользователь вычисляет ключ key = 26527036647708315 и передаёт 1-му пользователю. 1-й пользователь вычисляет
ключ key = 54454104531645922 и получает общий секретный ключ.
```

```
Базовый протокол (алгоритм Диффи-Хеллмана)
Количество пользователей: 2

Выбрать открытые параметры g и n вручную y/n? n
Открытый параметр g = 10133552624805800
Открытый параметр n = 55853486516845757

Пользователи будут выбирать секретные большие числа вручную u/n? y
1-й пользователь выбирает число a = 4568734568734587456334
2-й пользователь выбирает число b = 64786234586437856534

1-й пользователь вычисляет ключ key = 24993675384961553 и передаёт 2-му пользователю. 2-й пользователь вычисляет
key = 15731406598956895 и получает общий секретный ключ.
2-й пользователь вычисляет ключ key = 33080388367496307 и передаёт 1-му пользователю. 1-й пользователь вычисляет
ключ key = 15731406598956895 и получает общий секретный ключ.
```

```

Базовый протокол (алгоритм Диффи-Хеллмана)
Количество пользователей: 4

Выбрать открытые параметры g и n вручную y/n? n
Открытый параметр g = 160885037341690
Открытый параметр n = 1383376623562741

Пользователи будут выбирать секретные большие числа вручную y/n? n
1-й пользователь выбирает число a = 240711996
2-й пользователь выбирает число b = 363028680
3-й пользователь выбирает число c = 151410670
4-й пользователь выбирает число d = 200725308

1-й пользователь вычисляет ключ key = 1313655340252884 и передаёт 2-му пользователю
2-й пользователь вычисляет ключ key = 160885037341690 и передаёт 3-му пользователю
3-й пользователь вычисляет ключ key = 430867363159360 и передаёт 4-му пользователю
4-й пользователь вычисляет ключ key = 733363831837599 и получает общий секретный ключ

2-й пользователь вычисляет ключ key = 728861590259597 и передаёт 3-му пользователю
3-й пользователь вычисляет ключ key = 1132376312352652 и передаёт 4-му пользователю
4-й пользователь вычисляет ключ key = 159101963389173 и передаёт 1-му пользователю
1-й пользователь вычисляет ключ key = 733363831837599 и получает общий секретный ключ

3-й пользователь вычисляет ключ key = 430867363159360 и передаёт 4-му пользователю
4-й пользователь вычисляет ключ key = 733363831837599 и передаёт 1-му пользователю
1-й пользователь вычисляет ключ key = 1056444597621578 и передаёт 2-му пользователю
2-й пользователь вычисляет ключ key = 733363831837599 и получает общий секретный ключ

4-й пользователь вычисляет ключ key = 511964673711540 и передаёт 1-му пользователю
1-й пользователь вычисляет ключ key = 499298430714329 и передаёт 2-му пользователю
2-й пользователь вычисляет ключ key = 511964673711540 и передаёт 3-му пользователю
3-й пользователь вычисляет ключ key = 733363831837599 и получает общий секретный ключ

```

3 Листинг кода

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

using namespace std;
using namespace boost::multiprecision;
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)
        return lst[0].first;
    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
    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 nod(cpp_int y, cpp_int x) {
    cpp_int r = y % x;
    if (r == 0)
```

```

        return x;
    else
        return nod(x, r);
}

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

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;
}

```



```

pair <cpp_int, cpp_int> generateGN() {
    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 = rand() % 50 + 1;
        for (int i = 0; i < sBinSize; i++)
            sBin = sBin + to_string(rand() % 2);
        s = decForm(sBin);

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

    cpp_int a = 2;
    while (nDec > a) {
        cpp_int g = powClosed(a, nDec / q, n);
        if (g == 1) {
            a++;
            continue;
        }

        return make_pair(g, n);
    }
    return make_pair(0, 0); //Строка для обхода warning'a в Linux
}

void difHell2(cpp_int g, cpp_int n, vector <cpp_int> keys) {
    cpp_int keyForUser2 = powClosed(g, keys[0], n);
    cpp_int keyForUser1 = powClosed(g, keys[1], n);
    cout << "1-й пользователь вычисляет ключ key = " << keyForUser2 << " и
передает 2-му пользователю. ";
    cout << "2-й пользователь вычисляет key = " << powClosed(keyForUser2,
keys[1], n) << " и получает общий секретный ключ. \n";
    cout << "2-й пользователь вычисляет ключ key = " << keyForUser1 << " и
передает 1-му пользователю. ";
    cout << "1-й пользователь вычисляет ключ key = " <<
powClosed(keyForUser1, keys[0], n) << " и получает общий секретный ключ. \n";
}

void difHell(cpp_int g, cpp_int p, vector <cpp_int> keys) {
    int n = keys.size();
    for (int i = 0; i < n; i++) {
        cpp_int openKey = g;
        for (int j = 0; j < n; j++) {
            openKey = powClosed(openKey, keys[(j + i) % n], p);
            if (j == n - 1)
                cout << (j + i) % n + 1 << "-й пользователь
вычисляет ключ key = " << openKey << " и получает общий секретный ключ\n\n";
            else
                cout << (j + i) % n + 1 << "-й пользователь
вычисляет ключ key = " << openKey << " и передает " << (j + i + 1) % n + 1 << "-
му пользователю\n";
        }
    }
}

```

```

int main() {
    srand(time(0));
    setlocale(LC_ALL, "ru");
    cout << "\tБазовый протокол (алгоритм Диффи-Хеллмана) \n";

    int k;
    cout << "Количество пользователей: ";
    cin >> k;
    if (!k || k == 0 || k == 1) {
        cout << "Некорректный ввод данных\n";
        return 0;
    }

    pair <cpp_int, cpp_int> gn;
    char choice;
    cout << "\nВыбрать открытые параметры g и n вручную y/n? ";
    cin >> choice;
    while (true) {
        if (choice == 'y') {
            cout << "Открытый параметр g = ";
            cin >> gn.first;
            if (!gn.first) {
                cout << "Некорректный ввод данных\n";
                return 0;
            }
            if (funEuler(gn.first) == gn.first - 1)
                break;
            else {
                cout << gn.first << " не является простым
числом!\nОткрытый параметр g = ";
                cin >> gn.first;
            }

            cout << "Открытый параметр n = ";
            cin >> gn.second;
            if (!gn.second) {
                cout << "Некорректный ввод данных\n";
                return 0;
            }
            if (funEuler(gn.second) == gn.second - 1)
                break;
            else {
                cout << gn.second << " не является простым
числом!\nОткрытый параметр n = ";
                cin >> gn.second;
            }
        }
        else if (choice == 'n') {
            while (true) {
                gn = generateGN();
                if (gn.second > 1000000000000)
                    break;
            }
            cout << "Открытый параметр g = " << gn.first <<
"\nОткрытый параметр n = " << gn.second;
            break;
        }
        else {
            cout << "Некорректный ввод данных! \nВыбрать открытые
параметры g и n вручную y/n? ";
            cin >> choice;
        }
    }
}

```

```

    }
}

cout << "\n\n";
vector<cpp_int> keys(k);
cout << "Пользователи будут выбирать секретные большие числа вручную
y/n? ";
cin >> choice;
while (true) {
    if (choice == 'y') {
        for (int i = 0; i < k; i++) {
            cout << i + 1 << "-й пользователь выбирает число
" << char(97 + i) << " = ";
            cin >> keys[i];
        }
        break;
    }
    else if (choice == 'n') {
        for (int i = 0; i < k; i++) {
            keys[i] = (rand() * rand()) % gn.second;
            cout << i + 1 << "-й пользователь выбирает число
" << char(97 + i) << " = " << keys[i] << endl;
        }
        break;
    }
    else {
        cout << "Некорректный ввод данных! \nПользователи будут
выбирать секретные большие числа вручную y/n? ";
        cin >> choice;
    }
}

cout << endl;
if (k == 2)
    difHell2(gn.first, gn.second, keys);
else
    difHell(gn.first, gn.second, keys);
return 0;
}

```