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  профессор, д.ф.-м.н. | \_\_\_\_\_\_\_\_\_\_ | В. Е. Новиков |
|  | подпись, дата |  |

Саратов 2023

**СОДЕРЖАНИЕ**

1 Цель работы и порядок её выполнения…………………………………….3

2 Теория………………………………………………………………………..4

2.1 Обычный алгоритм Евклида………………………………….……4

2.2 Бинарный алгоритм Евклида………………………………………4

2.3 Расширенный алгоритм Евклида……………………………….…6

2.4 Греко-китайская теорема об остатках……………………………..7

2.5 Алгоритм Гарнера………………………………………………….8

2.6 Решение СЛУ методом Гаусса…………………………………….9

3 Результаты работы…………………………………………………..……..12

3.1 Оценки сложности рассмотренных алгоритмов………………...12

3.2 Результаты тестирования программ……………………………..12

3.3 Код программы……………………………………………………14

ЗАКЛЮЧЕНИЕ……………………………………..……………………….22

**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 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 mod *n*. Эти два целых числа хранить в секрете необязательно, Алиса и Боб могут договориться о6 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 *х* и посылает Бобу

*Х* = *gx* mod *n*

(2) Боб выбирает случайное большое целое число *у* и посылает Алисе

*Y* = *gy* mod *n*

(3) Алиса вычисляет

*k* = *Yx* mod *n*

(4) Боб вычисляет

*k*’ = *Xy* mod *n*

И *k*, и *k*’ равны *gxy* 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 *х* или *у*, 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 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 больную подгруппу мультипликативной группы mod *n*).

**2 Описание программы**

Программа, представленная ниже, содержит следующие функции:

• powClosed (*x*, *y*, *mod*) – возводит число *x* в степень *y* по модулю *mod*;

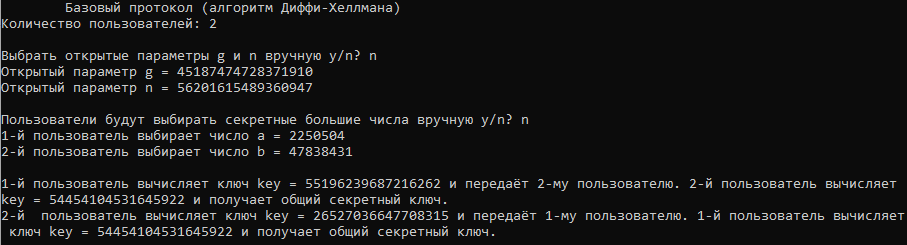
• miller\_rabin (*n*, *k* = 10) – проверка числа *n* на простоту с вероятностью ;

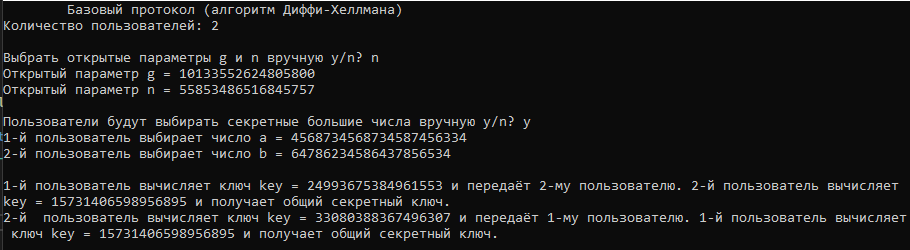
• generateGN() – генерация чисел *g* и *n* : *g* является первообразным корнем для *n*;

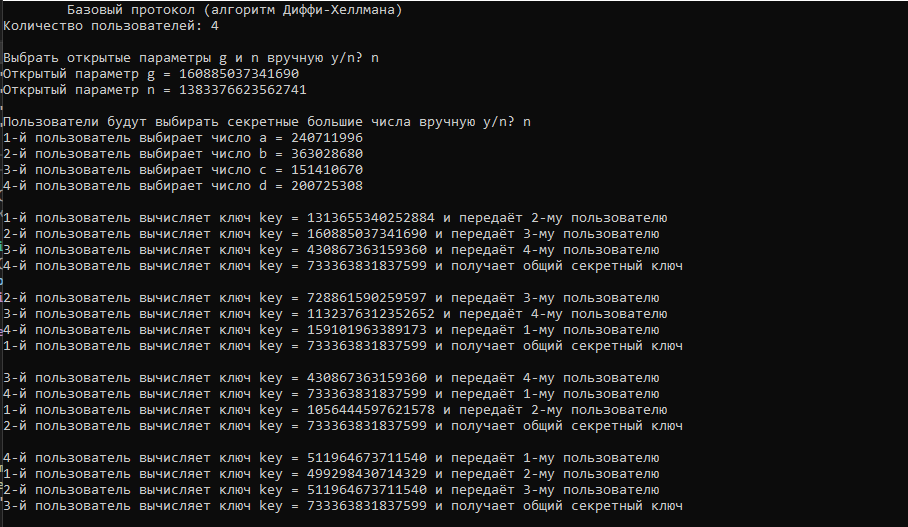
• diffHell2(*g*, *n*, *keys*) – генерация общего секретного ключа *gkeys*[0]•*keys*[1] (mod *n*)для двух пользователей;

• diffHell(*g*, *n*, *keys*) – генерация общего секретного ключа *gkeys*[0]•*keys*[1] •…•keys[*k*-1] (mod *n*)для *k* пользователей;

**2.1 Примеры работы программы**



****

****

**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>> xDegs;

for (int i = 0; i < lst.size(); i++)

xDegs.push\_back(make\_pair(x, lst[i]));

cpp\_int res = multMod(x, mod, xDegs);

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);//Cтрока для обхода warning'а 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 передаёт 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 передаёт 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 передаёт " << (j + i + 1) % n + 1 << "-му пользователю**\n**";

}

}

}

int main() {

srand(time(0));

setlocale(LC\_ALL, "ru");

cout << "**\t**Базовый 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 << "**\n**Выбрать 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!**\n**Открытый 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!**\n**Открытый параметр n = ";

cin >> gn.second;

}

}

else if (choice == 'n') {

while (true) {

gn = generateGN();

if (gn.second > 1000000000000)

break;

}

cout << "Открытый параметр g = " << gn.first << "**\n**Открытый параметр n = " << gn.second;

break;

}

else {

cout << "Неккоректный ввод данных! **\n**Выбрать 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 << "Неккоректный ввод данных! **\n**Пользователи 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;

}