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

**Протоколы анонимности**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Арбузова Матвея Александровича

|  |  |  |
| --- | --- | --- |
| Преподаватель  аспирант | \_\_\_\_\_\_\_\_\_\_ | Р. А. Фарахутдинов |
|  | подпись, дата |  |

Саратов 2023

**1 Постановка задачи**

Необходимо реализовать протокол голосования с одной центральной комиссией на базе протокола ANDOS.

**2 Теоретические сведения**

В криптографии протоколы тайного голосования – протоколы обмена данными для реализации безопасного тайного электронного голосования через интернет при помощи компьютеров, телефонов или других специальных вычислительных машин

Идеальный протокол тайного голосования должен обладать, по крайней мере, следующими шестью свойствами:

1) Участвовать в голосовании могут только граждане, имеющие право голоса;

2) Каждый избиратель может голосовать только один раз;

3) Никто не может установить, за кого проголосовал каждый избиратель;

4) Никто не может сделать дубликат бюллетеня с волеизъявлением любого избирателя;

5) Никто не может изменить результат любого избирателя;

6) Каждый избиратель может проверить, что его бюллетень учтён при подведении итогов голосования.

Кроме того, некоторые схемы голосования включают ещё одно требование:

7) Всем известно, кто участвовал в голосовании, а кто нет.

Протокол на основе ANDOS удовлетворяет шести первым требованиям идеального протокола, но не соответствует седьмому требованию.

**Протокол** **ANDOS**

Протокол раскрытия секретов «всё или ничего» (ANDOS) используется для того, чтобы передать одну из множества секретных строк, так, чтобы получатель мог получить только один секрет, а отправитель не знал, который именно.

Пусть Алиса обладает множеством секретов , Боб должен получить один секрет на его выбор , протокол не должен позволить бобу получить все секреты.

Такой протокол можно реализовать на основе криптосистемы RSA:

1) Алиса шифрует все секреты своим открытым ключом RSA: . И предоставляет к ним доступ Бобу.

2) Боб вычисляет , где – случайное число, меньшее чем , и отправляет Алисе;

3) Алиса вычисляет и передаёт Бобу;

4) Боб вычисляет .

Таким образом Боб получает секрет с выбранным им номером, и не может узнать ничего о других секретах. Алиса не может узнать ничего о номере секрета, который получил Боб.

**Протокол голосования с одной центральной комиссией на базе протокола ANDOS**

*Вход*: – число избирателей, – число претендентов.

*Выход*: Результаты голосования.

**1) Генерация основных параметров схемы:**

Шаг 1. Центральная избирательная комиссия (ЦИК) публикует список всех правомочных избирателей;

Шаг 2. Каждый избиратель сообщает ЦИК, намерен ли он голосовать;

Шаг 3. ЦИК публикует список избирателей, собирающихся принять участье в голосовании;

Шаг 4. Каждому избирателю отсылают по протоколу ANDOS идентификатор , при этом – простые, отличные друг от друга числа;

Шаг 5. Каждый избиратель генерирует пару открытый ключ/закрытый ключ: . Если обозначить выбор избирателя как , то избиратель создаёт и посылает в ЦИК следующее сообщение: , где – шифрование ключом . Данное сообщение отсылается анонимно;

Шаг 6. ЦИК подтверждает получение бюллетеня, публикуя с этой целью: ;

7) Каждый избиратель отправляет в ЦИК следующее сообщение: ;

8) ЦИК расшифровывает бюллетени. По окончанию выборов ЦИК публикует результаты, и, для каждого варианта голосования – список соответствующих значений .

На этапе 4 два избирателя могут получить одинаковые идентификаторы. Эту опасность можно свести к минимуму, если число допустимых идентификаторов значительно превышает число фактических избирателей. Если два избирателя пришлют бюллетени с одинаковыми идентификаторами , ЦИК генерирует новый идентификатор , отбирает одного из избирателей с одинаковыми идентификаторами и публикует пару . Тогда владелец этого бюллетеня узнает о произошедшей путанице и повторно отошлёт свой бюллетень с новым идентификатором, повторив этап 5 (повторно генерировать ключи не нужно).

Один из недостатков этого протокола заключается в том, что преступная ЦИК может воспользоваться бюллетенями избирателей, которые на втором этапе изъявили намерение голосовать, однако фактически не проголосовали. Ещё один недостаток – сложность протокола ANDOS (необходимо разбивать избирателей на небольшие группы).

**3 Практическая реализация**

**3.1 Описание программы**

Программа была написана на языке C++, и имеет множество функций.

Функция является точкой старта программы и отвечает за проверку корректности введённых, при запуске программы, числа избирателей , и числа претендентов .

Функция содержит все шаги описанного выше протокола, при этом для генерации простых чисел , получения модуля и ключей , системы RSA, получения ключей избирателей и используются функции из библиотеки .

В программе используются большие числа, работать с которыми позволяет подключённая библиотека , кроме того, силами данной библиотеки осуществляется генерация случайного числа из заданного диапазона в функции .

Для поиска обратного элемента и двух чисел в поле используется расширенный алгоритм Евклида – функция . Для проверки чисел на простоту используется алгоритм Миллера-Рабина, который реализован в функции . Для вывода ключей и шифртекста на экран используются функции , и . Шифрование и дешифрование реализовано в функциях и , а разобрать сообщение на части помогает функция . Кроме того, в программе реализована функция быстрого возведения в степень по модулю, которая используется в большом количестве шагов – функция .

**3.2 Результаты тестирования программы**

Отсутствие одного или нескольких параметров, при запуске программы, ведёт к соответствующей ошибке, данный запуск представлен на рисунке 1.

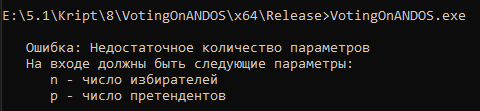


Рисунок 1 – Запуск программы без параметров

Числа и должны быть больше 0 – рисунок 2.

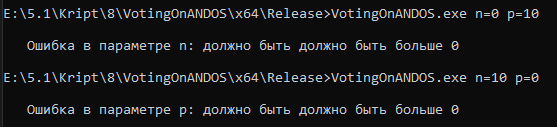


Рисунок 2 – Запуск программы с параметрами, не удовлетворяющими условиям

На рисунках 3-6 представлен успешный запуск программы для десяти разрешённых избирателей и пяти претендентов:

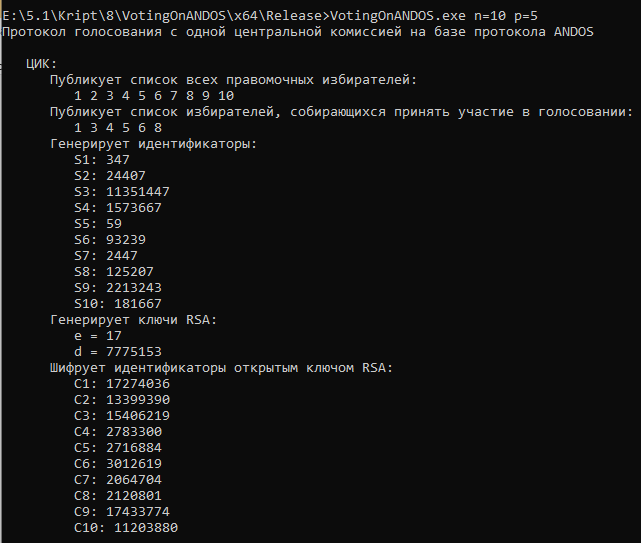


Рисунок 3 – Успешный запуск программы (Часть 1 – генерация и шифрование идентификаторов)



Рисунок 4 – Успешный запуск программы (Часть 2 – действия первого избирателя)



Рисунок 5 – Успешный запуск программы (Часть 3 – действия восьмого избирателя)

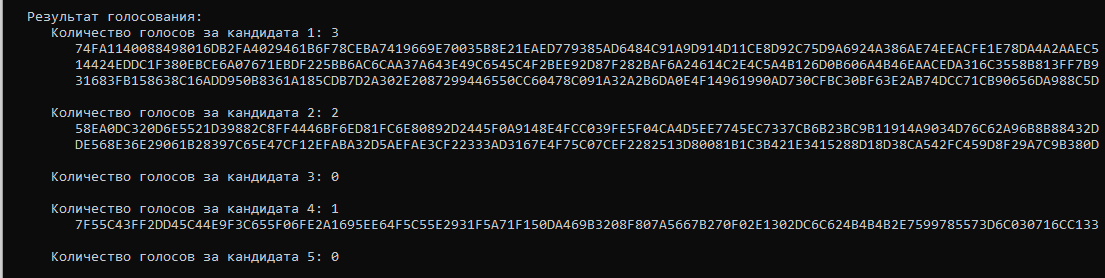


Рисунок 6 – Успешный запуск программы (Часть 4 – результат голосования)

**Листинг кода**

#include <iostream>

#include <string>

#include "osrng.h"

#include "dsa.h"

#include "dh.h"

#include <random>

#include "files.h"

#include <set>

#include <hex.h>

#include "rsa.h"

#include <boost/multiprecision/cpp\_int.hpp>

#include <boost/random/uniform\_int.hpp>

#include <boost/random/variate\_generator.hpp>

#include <boost/multiprecision/cpp\_dec\_float.hpp>

using namespace CryptoPP;

using namespace std;

using namespace boost::multiprecision;

using namespace boost::random;

AutoSeededRandomPool rng;

cpp\_int ModNegative(cpp\_int a, cpp\_int p) {

if (a < 0)

a = a + p \* (((-1 \* a) / p) + 1);

return a % p;

}

vector <cpp\_int> ExtendedEuclid(cpp\_int a, cpp\_int b) {

vector <cpp\_int> res(3);

if (a == 0) {

res = { b, 0, 1 };

return res;

}

vector <cpp\_int> c = ExtendedEuclid(b % a, a);

res = { c[0], c[2] - (b / a) \* c[1], c[1] };

return res;

}

cpp\_int Exponentiation(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;

}

bool MilRab(cpp\_int p, cpp\_int k) {

if (p == 1 || p == 2 || p == 3)

return true;

if (p % 2 == 0)

return false;

cpp\_int t = p - 1;

cpp\_int s = 0;

while (t % 2 == 0) {

t = t / 2;

s++;

}

for (cpp\_int i = 0; i < k; i++) {

cpp\_int a = rand() % (p - 3) + 2;

if (ExtendedEuclid(p, a)[0] > 1)

return false;

cpp\_int x = Exponentiation(a, t, p);

if (x == 1 || x == p - 1)

continue;

for (cpp\_int g = 1; g < s; g++) {

x = x \* x % p;

if (x == 1)

return false;

if (x == p - 1)

break;

}

if (x != p - 1)

return false;

}

return true;

}

cpp\_int Random(cpp\_int minim, cpp\_int maxim) {

random\_device gen;

boost::random::uniform\_int\_distribution<cpp\_int> ui(minim, maxim);

return ui(gen);

}

cpp\_int IntegerToCppint(const Integer number) {

ostringstream oss;

oss << number;

string str(oss.str());

str.erase(str.size() - 1, 1);

cpp\_int res(str);

return res;

}

string Encryption(RSA::PublicKey key, string message) {

string res;

RSAES\_OAEP\_SHA\_Encryptor e(key);

StringSource ss1(message, true,

new PK\_EncryptorFilter(rng, e,

new StringSink(res)

)

);

return res;

}

string Decryption(RSA::PrivateKey key, string message) {

string res;

RSAES\_OAEP\_SHA\_Decryptor d(key);

StringSource ss2(message, true,

new PK\_DecryptorFilter(rng, d,

new StringSink(res)

)

);

return res;

}

vector <string> UnMess(string mess, int k) {

string find = " ^ ", prom, resStr;

vector <string> resVec;

while (k != 0) {

prom = { mess[0] , mess[1], mess[2] };

if (prom == find) {

mess.erase(0, 3);

resVec.push\_back(resStr);

resStr.clear();

k--;

}

else {

resStr = resStr + mess[0];

mess.erase(0, 1);

}

}

resVec.push\_back(mess);

return resVec;

}

void PrintRes(string str) {

HexEncoder encoder(new FileSink(cout));

encoder.Put((const byte\*)&str[0], str.size());

encoder.MessageEnd();

}

void Encode(const BufferedTransformation& bt){

HexEncoder encoder(new FileSink(cout));

bt.CopyTo(encoder);

encoder.MessageEnd();

}

void EncodePublicKey(const RSA::PublicKey& key){

ByteQueue queue;

key.DEREncodePublicKey(queue);

Encode(queue);

}

void EncodePrivateKey(const RSA::PrivateKey& key) {

ByteQueue queue;

key.DEREncodePrivateKey(queue);

Encode(queue);

}

void CountingOfVotes(vector< vector<pair <cpp\_int, string>>> bulletin, vector< pair <cpp\_int, RSA::PrivateKey>> keys, int p) {

vector< vector <string>> result(p);

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

if (bulletin[i].size() == 2) {

if (bulletin[i][1].first == keys[i].first) {

string messDe2 = Decryption(keys[i].second, bulletin[i][1].second);

vector <string> SbAndV2 = UnMess(messDe2, 1);

cpp\_int SbDe2(SbAndV2[0]);

int choiceDe2 = stoi(SbAndV2[1]);

if (SbDe2 == bulletin[i][1].first) {

string messDe1 = Decryption(keys[i].second, bulletin[i][0].second);

vector <string> SbAndV1 = UnMess(messDe1, 1);

cpp\_int SbDe1(SbAndV1[0]);

int choiceDe1 = stoi(SbAndV1[1]);

if (SbDe1 == bulletin[i][0].first)

result[choiceDe2 - 1].push\_back(bulletin[i][1].second);

}

}

}

else {

if (bulletin[i][0].first == keys[i].first) {

string messDe = Decryption(keys[i].second, bulletin[i][0].second);

vector <string> SbAndV = UnMess(messDe, 1);

cpp\_int SbDe(SbAndV[0]);

int choiceDe = stoi(SbAndV[1]);

if (SbDe == bulletin[i][0].first)

result[choiceDe - 1].push\_back(bulletin[i][0].second);

}

}

}

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

cout << "\n Количество голосов за кандидата " << i + 1 << ": " << result[i].size();

for (int j = 0; j < result[i].size(); j++) {

cout << "\n ";

PrintRes(result[i][j]);

}

cout << "\n";

}

}

void VotingOnANDOS(int n, int p) {

//Публикация всех возможных избирателей

cout << "\n ЦИК:";

cout << "\n Публикует список всех правомочных избирателей:\n ";

for (int i = 0; i < n; i++)

cout << i + 1 << " ";

//Избиратели, которые хотят голосовать

cout << "\n Публикует список избирателей, собирающихся принять участие в голосовании:\n ";

vector <int> voters;

for (int i = 0; i < n; i++)

if (rand() % 2 == 1) {

voters.push\_back(i);

cout << i + 1 << " ";

}

//Генерация идентификаторов (простые числа)

set<cpp\_int> setS;

vector<cpp\_int> vecS;

DH dh;

while (setS.size() != n) {

int sizeS = setS.size();

dh.AccessGroupParameters().GenerateRandomWithKeySize(rng, rand() % 21 + 4);

cpp\_int prom = IntegerToCppint(dh.GetGroupParameters().GetModulus());

setS.insert(prom);

if (setS.size() != sizeS)

vecS.push\_back(prom);

}

cout << "\n Генерирует идентификаторы:\n";

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

cout << " S" << i + 1 << ": " << vecS[i] << "\n";

//Генерация ключей RSA для ЦИК

InvertibleRSAFunction params;

params.GenerateRandomWithKeySize(rng, 25);

cpp\_int nA = IntegerToCppint(params.GetModulus());

cpp\_int eA = IntegerToCppint(params.GetPublicExponent());

cpp\_int dA = IntegerToCppint(params.GetPrivateExponent());

cout << " Генерирует ключи RSA:\n e = " << eA << "\n d = " << dA;

//Шифрование идентификаторов открытым ключом RSA

vector<cpp\_int> vecC;

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

vecC.push\_back(Exponentiation(vecS[i], eA, nA));

cout << "\n Шифрует идентификаторы открытым ключом RSA:\n";

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

cout << " C" << i + 1 << ": " << vecC[i] << "\n";

//Голосование

set<cpp\_int> usedS;

vector< vector<pair <cpp\_int, string>>> bulletin;

vector< pair <cpp\_int, RSA::PrivateKey>> keys;

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

//Получение идентификатора

cout << "\n\n Голосующий " << voters[i] + 1 << ":\n";

int b = rand() % n;

cout << " Выбрал C" << b + 1 << ": " << vecC[b];

cpp\_int r = Random(2, nA - 1);

while (ExtendedEuclid(r, nA)[0] != 1)

r = Random(1, nA - 1);

cout << "\n Выбрал случайное число r = " << r;

cpp\_int Csh = (vecC[b] \* Exponentiation(r, eA, nA)) % nA;

cout << "\n Вычислил и отправил ЦИК значение С' = С" << b + 1 << " \* r^e (mod n) = " << Csh;

cout << "\n\n ЦИК:";

cpp\_int Psh = Exponentiation(Csh, dA, nA);

cout << "\n Вычислил и отправил голосующему P' = С'^d (mod n) = " << Psh;

cout << "\n\n Голосующий " << voters[i] + 1 << ":\n";

cpp\_int Sb = (Psh \* ModNegative(ExtendedEuclid(r, nA)[1], nA)) % nA;

cout << " Получил идентификатор S" << b + 1 << " = " << Sb << " с помощью выражения P' \* r^-1 (mod n)";

//генерация ключей голосующим

params.GenerateRandomWithKeySize(rng, 512);

RSA::PrivateKey d(params);

RSA::PublicKey k(params);

cout << "\n Сгенерировал открытый ключ k:\n ";

EncodePublicKey(k);

cout << "\n Сгенерировал закрытый ключ d:\n ";

EncodePrivateKey(d);

//Выбор кандидата

int choice = rand();

choice = choice % p + 1;

cout << "\n Выбрал кандидата: " << choice;

//Шифрование и отправка сообщения

string mess = to\_string(Sb) + " ^ " + to\_string(choice);

string messEn = Encryption(k, mess);

cout << "\n Отправил ЦИК пару (S" << b + 1 << ", Ek(S" << b + 1 << ", v)):\n (" << Sb << ", ";

PrintRes(messEn);

cout << ")";

//Подтверждение получения

int sizeUS = usedS.size();

usedS.insert(Sb);

bool checker = true;

bulletin.push\_back({ make\_pair(Sb, messEn) });

//Идентификатр совпал

if (sizeUS == usedS.size()) {

//Новый идентификатор

int sizeS = setS.size();

cpp\_int Ssh;

while (setS.size() == sizeS) {

dh.AccessGroupParameters().GenerateRandomWithKeySize(rng, rand() % 21 + 4);

Ssh = IntegerToCppint(dh.GetGroupParameters().GetModulus());

setS.insert(Ssh);

}

cout << "\n\n ЦИК публикует пару (S', Ek(S" << b + 1 << ", v), так как выбранный идентификатор занят:\n (" << Ssh << ", ";

PrintRes(messEn);

cout << ")";

//Новая отправка голосующего

cout << "\n\n Голосующий " << voters[i] + 1 << ":";

mess = to\_string(Ssh) + " ^ " + to\_string(choice);

messEn = Encryption(k, mess);

cout << "\n Отправил ЦИК пару (S', Ek(S', v)):\n (" << Ssh << ", ";

PrintRes(messEn);

cout << ")";

Sb = Ssh;

usedS.insert(Sb);

bulletin[bulletin.size()-1].push\_back(make\_pair(Sb, messEn));

checker = false;

}

//Теперь идентификатор в порядке

if (checker)

cout << "\n\n ЦИК публикует Ek(S" << b + 1 << ", v):\n ";

else

cout << "\n\n ЦИК публикует Ek(S', v):\n ";

PrintRes(messEn);

//Голосующий отправляет ключ

if (checker)

cout << "\n\n Голосующий " << voters[i] + 1 << " отправил ЦИК пару (S" << b + 1 << ", d):\n (" << Sb << ", ";

else

cout << "\n\n Голосующий " << voters[i] + 1 << " отправил ЦИК пару (S', d):\n (" << Sb << ", ";

EncodePrivateKey(d);

cout << ")\n";

keys.push\_back(make\_pair(Sb, d));

}

//Результат голосования

cout << "\n\n Результат голосования:";

CountingOfVotes(bulletin, keys, p);

}

void ErrMess() {

cerr << " На входе должны быть следующие параметры:\n"

<< " n - число избирателей \n"

<< " p - число претендентов\n";

}

int main(int argc, char\*\* argv) {

setlocale(LC\_ALL, "Russian");

srand(time(NULL));

if (argc < 3) {

cerr << "\n Ошибка: Недостаточное количество параметров\n";

ErrMess();

return 0;

}

int n = -1, p = -1;

string prom1, prom2;

for (int i = 1; i < argc; i++) {

prom1 = argv[i];

try {

prom1.erase(2, prom1.size() - 2);

if (prom1 == "n=") {

prom2 = argv[i];

prom2.erase(0, 2);

try {

n = stoi(prom2);

}

catch (exception) {

cerr << "\n Ошибка в параметре n: передано некорректное число\n";

ErrMess();

return 0;

}

continue;

}

if (prom1 == "p=") {

prom2 = argv[i];

prom2.erase(0, 2);

try {

p = stoi(prom2);

}

catch (exception) {

cerr << "\n Ошибка в параметре p: передано некорректное число\n";

ErrMess();

return 0;

}

continue;

}

}

catch (exception) {

continue;

}

}

if (n < 0 || p < 0) {

cerr << "\n Ошибка: некоторые параметры отсутствуют или пустые\n";

ErrMess();

return 0;

}

if (n < 1) {

cerr << "\n Ошибка в параметре n: должно быть должно быть больше 0\n";

return 0;

}

if (p < 1) {

cerr << "\n Ошибка в параметре p: должно быть должно быть больше 0\n";

return 0;

}

cout << "Протокол голосования с одной центральной комиссией на базе протокола ANDOS\n";

VotingOnANDOS(n, p);

return 0;

}