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 $S_1, S_2, \dots, S_k$, Боб должен получить один секрет на его выбор S_b, 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: $C_i = S_i^e (mod\ n)$. И предоставляет к ним доступ Бобу.

2) Боб вычисляет $C' = C_b r^e (mod\ n)$, где r – случайное число, меньшее чем n, и отправляет Алисе;

3) Алиса вычисляет $P' = C'^d (mod\ n)$ и передаёт Бобу;

4) Боб вычисляет $S_b = P' r^{-1} (mod\ n)$.

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: n – число избирателей, p – 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 S_b, при этом $S_1, \dots S_n$ – простые, отличные друг от друга числа;

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

Шаг 6. ЦИК подтверждает получение бюллетеня, публикуя с этой целью: $E_k(S_b, choice)$;

7) Каждый избиратель отправляет в ЦИК следующее сообщение: (S_b, d);

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

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 S_{same}, ЦИК генерирует новый идентификатор S', отбирает одного из избирателей с одинаковыми идентификаторами и публикует пару $(S', E_k(S_{same}, choice))$. 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 $main$ является точкой старта программы и отвечает за проверку корректности введённых, при запуске программы, числа избирателей n, и числа претендентов p.

Функция $VotingOnANDOS$ содержит все шаги описанного выше протокола, при этом для генерации простых чисел $S_1, \dots, S_n$, получения модуля

и ключей e, d системы RSA, получения ключей избирателей k и d используются функции из библиотеки $Crypto++$.

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

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

3.2 Результаты тестирования программы

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

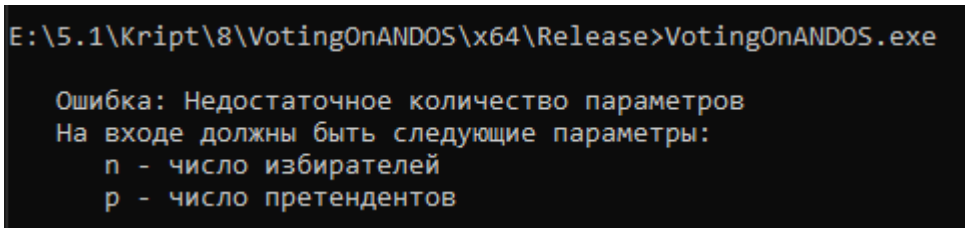

Рисунок 1 – Запуск программы без параметров

Числа n и p должны быть больше 0 – рисунок 2.

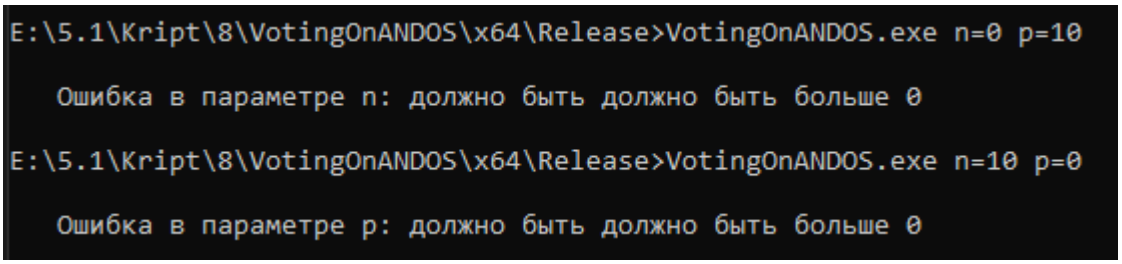

Рисунок 2 – Запуск программы с параметрами, не удовлетворяющими условиям

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

```
E:\5.1\Kript\8\VotingOnANDOS\x64\Release>VotingOnANDOS.exe n=10 p=5
Протокол голосования с одной центральной комиссией на базе протокола ANDOS

   ЦИК:
      Публикует список всех правомочных избирателей:
         1 2 3 4 5 6 7 8 9 10
      Публикует список избирателей, собирающихся принять участие в голосовании:
         1 3 4 5 6 8
      Генерирует идентификаторы:
         S1: 347
         S2: 24407
         S3: 11351447
         S4: 1573667
         S5: 59
         S6: 93239
         S7: 2447
         S8: 125207
         S9: 2213243
         S10: 181667
      Генерирует ключи RSA:
         e = 17
         d = 7775153
      Шифрует идентификаторы открытым ключом RSA:
         C1: 17274036
         C2: 13399390
         C3: 15406219
         C4: 2783300
         C5: 2716884
         C6: 3012619
         C7: 2064704
         C8: 2120801
         C9: 17433774
         C10: 11203880
```

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

```
   Голосующий 1:
      Выбрал C10: 11203880
      Выбрал случайное число r = 8162811
      Вычислил и отправил ЦИК значение C' = C10 * r^e (mod n) = 249503

   ЦИК:
      Вычислил и отправил голосующему P' = C'^d (mod n) = 2645800

   Голосующий 1:
      Получил идентификатор S10 = 181667 с помощью выражения P' * r^-1 (mod n)
      Сгенерировал открытый ключ k:
         3046024100B3EBFA815303887A419BBD45E6A57475530 1326AB8126F001D7CA29BA60D6B4F9E8376545BE51F33006193583F709145D97E3570F3CF437F08E674536C9F7BE9020111
      Сгенерировал закрытый ключ d:
         3082013A020100024100B3EBFA815303887A419BBD45E6A57475530 1326AB8126F001D7CA29BA60D6B4F9E8376545BE51F33006193583F709145D97E3570F3CF437F08E674536C9F7BE
90201102400BE816397FD8B453627599B3AF436B78B48D41743D1F5667897B124A4CDF01743725AF689CEFA68C46D457A814B266E7D04B3D2D58A6F6E996A1A5617AACE49102210 0FB051FA25EA
E808F9A715987C8A82B45C04558C7607008F137E30058BB147FA1022100B77DC32FD62DAD781B2E9B72CC408BDF5BE45CA70267C3E07B070639ACABB749022100A26CBA1DC4CB4420AF3A48FD81D
63A1E12FFB1EA6B93C98D060B5A93C4588ED1022100971C4663A152CB179DEA25A9D56255124BAD010201FB19C7ECD89BB706AB87E10221009DFCB609AAAE29B56A3805778443CFC0A971CBE69C3
45859A07DA4DB1A2634C7
      Выбрал кандидата: 2
      Отправил ЦИК пару (S10, Ek(S10, v)):
         (181667, 58EA0DC320D6E5521D39882C8FF4446BF6ED81FC6E80892D2445F0A9148E4FCC039FE5F04CA4D5EE7745EC7337CB6B23BC9B11914A9034D76C62A96B8B88432D)

   ЦИК публикует Ek(S10, v):
      58EA0DC320D6E5521D39882C8FF4446BF6ED81FC6E80892D2445F0A9148E4FCC039FE5F04CA4D5EE7745EC7337CB6B23BC9B11914A9034D76C62A96B8B88432D

   Голосующий 1 отправил ЦИК пару (S10, d):
      (181667, 3082013A020100024100B3EBFA815303887A419BBD45E6A57475530 1326AB8126F001D7CA29BA60D6B4F9E8376545BE51F33006193583F709145D97E3570F3CF437F08E674536
C9F7BE90201102400BE816397FD8B453627599B3AF436B78B48D41743D1F5667897B124A4CDF01743725AF689CEFA68C46D457A814B266E7D04B3D2D58A6F6E996A1A5617AACE49102210 0FB051
FA25EAE808F9A715987C8A82B45C04558C7607008F137E30058BB147FA1022100B77DC32FD62DAD781B2E9B72CC408BDF5BE45CA70267C3E07B070639ACABB749022100A26CBA1DC4CB4420AF3A4
8FD81D63A1E12FFB1EA6B93C98D060B5A93C4588ED1022100971C4663A152CB179DEA25A9D56255124BAD010201FB19C7ECD89BB706AB87E10221009DFCB609AAAE29B56A380577844 3CFC0A971C
BE69C345859A07DA4DB1A2634C7)
```

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

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

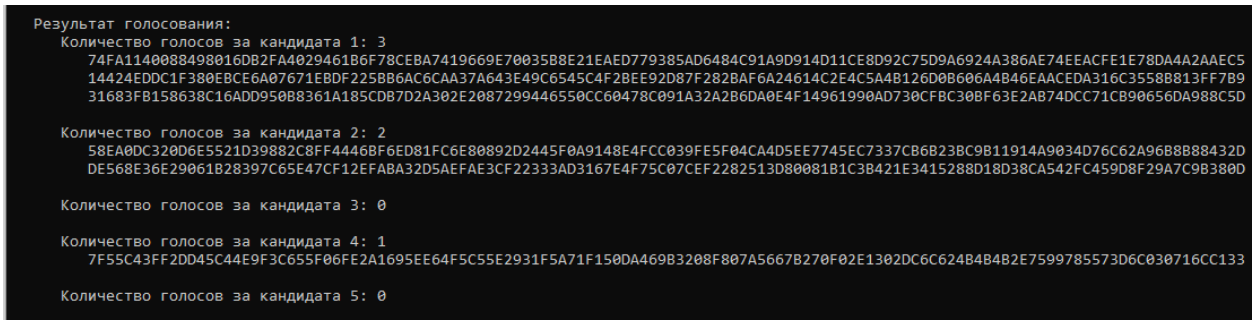

Рисунок 6 – Успешный запуск программы (Часть 4 – результат голосования)

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 C' = C" << 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' = C'^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;
      }
```