

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 обмена ключами

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

подпись, дата

Р. А. Фарахутдинов

Саратов 2023

1 Постановка задачи

Необходимо реализовать протокол Ньюмана-Стаблбайна, являющегося протоколом обмена ключами.

2 Теоретические сведения

Из-за недостатков системы или саботажа синхронизация часов может быть нарушена. Если часы сбиваются, против большинства протоколов может быть использован определённый способ вскрытия. Если часы отправителя опережают часы получателя, Мэллори может перехватить сообщение отправителя и повторно использовать его позднее, когда метка времени станет текущей в месте нахождения получателя. Этот способ, называющийся вскрытием с подавлением повторной передачи, может привести к неприятным последствиям.

Протокол Neuman-Stubblebine пытается противостоять вскрытию с подавлением повторной передачи. Этот отличный протокол является улучшением Yahalom.

Протокол Neuman-Stubblebine

Вход: Целое число l – битовая длина случайных чисел Алисы и Боба.

Выход: K – секретный ключ.

Шаг 1. Алиса объединяет своё имя A и случайное число R_A , после чего отправляет созданное сообщение (A, R_A) Бобу;

Шаг 2. Боб объединяет имя Алисы, её случайное число и метку времени T_B , шифрует созданное сообщение общим с Трентом ключом E_B и посылает его Тренту, добавляя своё имя B и новое случайное число R_B (отправляет $(B, R_B, E_B(A, R_A, T_B))$);

Шаг 3. Трент генерирует случайный сеансовый ключ K . Затем он создаёт два сообщения. Первое включает имя Боба, случайное число Алисы, случайный сеансовый ключ, метку времени и шифрует ключом, общим для Трента и Алисы E_A . Второе состоит из имени Алисы, сеансового ключа, метки времени и шифруется ключом, общим для Трента и Боба. Трент посылает оба

сообщения $(E_A(B, R_A, K, T_B))$ и $(E_B(A, K, T_B))$ Алисе вместе со случайным числом боба R_B ;

Шаг 4. Алиса расшифровывает сообщение, зашифрованное её ключом, извлекает K и убеждается, что R_A совпадает со значением, отправленным на шаге (1). Алиса посылает Бобу два сообщения. Одним является сообщение Трента зашифрованное ключом Боба $(E_B(A, K, T_B))$. Второе – это R_B , зашифрованное сеансовым ключом $(E_k(R_B))$;

Шаг 5. Боб расшифровывает сообщение, зашифрованное его ключом, извлекает K и убеждается, что значения T_B и R_B те же, что и отправленные на шаге (2).

Если оба случайных числа и метка времени совпадают, Алиса и Боб убеждаются в подлинности друг друга и получают секретный ключ. Синхронизация часов не требуется, так как метка времени определяется только по часам Боба, и только Боб проверяет созданную им метку времени.

У этого протокола есть ещё одно полезное свойство – Алиса может использовать полученное от Трента сообщение для последующей проверки подлинности Боба в пределах некоторого времени. Предположим, что Алиса и Боб выполнили приведённый выше протокол, провели и завершили сеанс связи. Алиса и Боб могут повторно проверить подлинность друг друга, не обращаясь к Тренту.

Шаг 1. Алиса посылает Бобу сообщение $(E_B(A, K, T_B))$, присланное ей Трентом на шаге (3) и новое случайное число R'_A ;

Шаг 2. Боб посылает Алисе другое новое случайное число R'_B и случайное число, присланное Алисой, шифруя его сеансовым ключом связи $E_K(R'_A)$;

Шаг 3. Алиса посылает Бобу его новое случайное числа, шифруя его сеансовым ключом связи $E_K(R'_B)$.

Новые случайные числа защищают от вскрытия с повторной передачей.

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 введенной, при запуске программы, длинны случайных чисел.

Функция *NeumanStubblebine* содержит все шаги описанного выше алгоритма, при этом для генерации ключей E_A , E_B , K используются функции из библиотеки *Crypto++*, а функции *FirstPass*, *SecondPass*, *ThirdPass*, *FourthPass*, *FifthPass*, *Check1Pass*, *Check2Pass*, *Check3Pass*, *Check4Pass* разделяют шаги алгоритма на проходы, в каждом из которых вычисляются необходимые значения.

За шифрование и расшифрование отвечают функции *Encryption* и *Decryption*, использующие для своей работы шифр *AES* из библиотеки *Crypto++*.

В программе используются большие числа, работать с которыми позволяет подключённая библиотека *boost*, кроме того, силами данной библиотеки осуществляется генерация случайных чисел Алисы и Боба в функции *Random*.

Разделение сообщений на составляющие происходит с помощью функции *UnMess*. Вывод результатов на экран осуществляется в шестнадцатеричном формате с помощью функции *PrintRes*.

3.2 Результаты тестирования программы

При запуске программы без параметров выведет соответствующую ошибку, данный запуск представлен на рисунке 1.

```
C:\Windows\system32\cmd.exe
E:\>E:\5.1\Kript\2\NS1\x64\Release\NS1.exe
Необходимо ввести 1 - битовую длину случайных чисел Алисы и Боба
E:\>
```

Рисунок 1 – Запуск программы без параметров

Ввод случайного набора символов, приводит к ошибке – рисунок 2.

```
C:\Windows\system32\cmd.exe
E:\>E:\5.1\Kript\2\NS1\x64\Release\NS1.exe e212wqasag
Число 1 должно быть целым
```

Рисунок 2 – Запуск программы со случайным набором символов в качестве параметра

Кроме того, l должен быть больше 1 – рисунок 3.

```
C:\Windows\system32\cmd.exe
E:\>E:\5.1\Kript\2\NS1\x64\Release\NS1.exe -4
Число 1 должно быть больше 1
```

Рисунок 3 – Запуск программы с параметром меньшим единицы

На рисунках 4-7 представлены успешные запуски программ.

```
C:\Windows\system32\cmd.exe
E:\>E:\5.1\Kript\2\NS1\x64\Release\NS1.exe 10
Протокол Neuman-Stubblebine

Алиса генерирует случайное число Ra = 383637
И отправляет Бобу сообщение (A, Ra) = 416C696365205E20383637

Метка времени Tb = 31363937363330363233
Боб генерирует случайное число Rb = 373630
И отправляет Тренту сообщение (B, Rb, Eb(A, Ra, Tb))
426F62205E20373630205E20F7D52A88557E85B3E23D5EB6CE35F2F71B3FBFC1C7FF56B92B9F421877E16CC6

Трент генерирует сеансовый ключ K = 4EEC905B8DB50ABA597808EB0EBA0108
И посылает Алисе следующие сообщения:
Ea(B, Ra, K, Tb) = 9EEEE995B62356C4FCE2E3319EC8CF420C819736B26338F3EF61448F7282CF63D9B1D26A0F7C3418FFDAA32902CA1D27
Eb(A, K, Tb) = EBE055428C6C4B6CD2C91CB30396E8F69988EA2207C59952E10857D2B99E391FF98F9D34838287231989A3187CE130D2
Rb = 373630

Алиса извлекает из полученного сообщения:
ключ K = 4EEC905B8DB50ABA597808EB0EBA0108
свое секретное число Ra = 383637
(Случайное число Алисы полученное из сообщения и полученное на первом проходе совпадает)
Алиса посылает Бобу следующие сообщения:
Eb(A, K, Tb) = EBE055428C6C4B6CD2C91CB30396E8F69988EA2207C59952E10857D2B99E391FF98F9D34838287231989A3187CE130D2
Ek(Rb) = 9E1FD81AB7F7031D41FC522EA7ECB63A

Боб извлекает из полученных сообщений:
ключ K = 4EEC905B8DB50ABA597808EB0EBA0108
метку времени Tb = 31363937363330363233
свое секретное число Rb = 373630
(Временная метка полученная из сообщения и полученная на втором проходе совпадает)
(Случайное число Боба полученное из сообщения и полученное на втором проходе совпадает)

Ключи, полученные Алисой и Бобом, совпали
```

Рисунок 4 – Запуск программы с параметром $l = 10$, первая часть вывода

```
C:\Windows\system32\cmd.exe
Повторная проверка:
Алиса отправляет Бобу сообщения:
  Eb(A, K, Tb) = EBE055428C6C4B6CD2C91CB30396E8F69988EA2207C59952E10857D2B99E391F98F9D34838287231989A3187CE130D2
  R'a = 373037

Боб отправляет Алисе сообщения:
  R'b = 373937
  Ek(R'a) = 9D1D9CF07541DBE4BADB546FA966D208

Алиса извлекает:
  R'a = 373037
И отправляет Бобу сообщение:
  Ek(R'b) = 60181013375490D4656AB1E22959C627

Боб извлекает:
  R'b = 373937

Полученное и отправленное R'a совпадает
Полученное и отправленное R'b совпадает
```

Рисунок 5 – Вторая часть вывода

```
C:\Windows\system32\cmd.exe
E:\>E:\5.1\Kript\2\NS1\x64\Release\NS1.exe 128
Протокол Neuman-Stubblebine

Алиса генерирует случайное число Ra = 313730343334343834343131343632383836323131313037373638303234353531303938353730
И отправляет Бобу сообщение (A, Ra) = 416C696365205E20313730343334343834343131343632383836323131313037373638303234353531
303938353730

Метка времени Tb = 31363937363330383338
Боб генерирует случайное число Rb = 323035363532383734313034363436373138333436393633383733343330393631363335333439
И отправляет Тренту сообщение (B, Rb, Eb(A, Ra, Tb))
  426F62205E20323035363532383734313034363436373138333436393633383733343330393631363335333439205E208C8C41C14CC61B350C4D4
FD4AFA21F4E91B860F1C83E702EADCC0B4E093DDE1CB506F8ECE1A4C0E47F4E2F6009EB0D8CF9E6ED3C33FEFF3F3A6978D0FF9D477B

Трент генерирует сеансовый ключ K = 514F1ADCCE38E7141F30A24E5302745E
И посылает Алисе следующие сообщения:
  Ea(B, Ra, K, Tb) = EB2CCE40F0E0F68ADF94661AD7C63106A8D21EB86CCDACCDDF5A763F05B5D37A769F1B33E722868C438DD7658560A2F29
5FD2C57AE795DEEC6D02B90E60856CBAF2CF174EB24B889AD197409BDF3CBC
  Eb(A, K, Tb) = 0E6242EBD1CA8A8738E94F5FB1C86716C3EA082B161F14A23D4E57FCC57F4995D9B217808A3184A59A69803282DAAEF1
  Rb = 323035363532383734313034363436373138333436393633383733343330393631363335333439

Алиса извлекает из полученного сообщения:
  ключ K = 514F1ADCCE38E7141F30A24E5302745E
  своё секретное число Ra = 313730343334343834343131343632383836323131313037373638303234353531303938353730
  (Случайное число Алисы полученное из сообщения и полученное на первом проходе совпадает)
Алиса посылает Бобу следующие сообщения:
  Eb(A, K, Tb) = 0E6242EBD1CA8A8738E94F5FB1C86716C3EA082B161F14A23D4E57FCC57F4995D9B217808A3184A59A69803282DAAEF1
  Ek(Rb) = F93B34CA9845E3CD6CB058B48B183D91E6730BA506B357A0724A77CEF80FB16DE30E16247A790BA0F2962BF883185BFB

Боб извлекает из полученных сообщений:
  ключ K = 514F1ADCCE38E7141F30A24E5302745E
  метку времени Tb = 31363937363330383338
  своё секретное число Rb = 323035363532383734313034363436373138333436393633383733343330393631363335333439
  (Временная метка полученная из сообщения и полученная на втором проходе совпадает)
  (Случайное число Боба полученное из сообщения и полученное на втором проходе совпадает)

Ключи, полученные Алисой и Бобом, совпали
```

Рисунок 6 – Запуск программы с параметром $l = 128$, первая часть вывода

```
Повторная проверка:
Алиса отправляет Бобу сообщения:
  Eb(A, K, Tb) = 0E6242EBD1CA8A8738E94F5FB1C86716C3EA082B161F14A23D4E57FCC57F4995D9B217808A3184A59A69803282DAAEF1
  R'a = 323638353230353135363137333033313630303235343638383139393135373537333837323130

Боб отправляет Алисе сообщения:
  R'b = 323333313033333832353530323436353532313134383832393937383839373235333235313336
  Ek(R'a) = D0C56938617F3425F94776C78680D5286EDECA9C32BA347290359D2EA3625A1F46696AAAA83177703766F4A691481B24

Алиса извлекает:
  R'a = 323638353230353135363137333033313630303235343638383139393135373537333837323130
И отправляет Бобу сообщение:
  Ek(R'b) = 5DE05E45882A560055C3575C8DB1090659DEDC68BD13FCA1FA6762F501907E46F86BDA5F83CA84D1431A5859B879C9B6

Боб извлекает:
  R'b = 323333313033333832353530323436353532313134383832393937383839373235333235313336

Полученное и отправленное R'a совпадает
Полученное и отправленное R'b совпадает
```

Рисунок 7 – Вторая часть вывода

Листинг кода

```
#include "cryptlib.h"
#include "rijndael.h"
#include "modes.h"
#include "files.h"
#include "osrng.h"
#include "hex.h"

#include <iostream>
#include <string>
#include <random>
#include <boost/multiprecision/cpp_int.hpp>
#include <boost/random/mersenne_twister.hpp>
#include <boost/random/uniform_int.hpp>
#include <boost/random/variante_generator.hpp>
#include <vector>
#include <time.h>

using namespace std;
using namespace boost::multiprecision;
using namespace boost::random;
using namespace CryptoPP;

cpp_int minim, maxim;
AutoSeededRandomPool prng;
SecByteBlock iv(AES::BLOCKSIZE);

pair <cpp_int, cpp_int> MinMax(int l) {
    cpp_int deg = 2;
    cpp_int maximum = 1;
    for (int i = 2; i <= l; i++) {
        maximum = maximum + deg;
        deg = deg * 2;
    }
    return make_pair(deg / 2, maximum);
}

cpp_int Random() {
    random_device gen;
    boost::random::uniform_int_distribution<cpp_int> ui(minim, maxim);
    return ui(gen);
}

string Encryption(SecByteBlock key, string message) {
    CBC_Mode< AES >::Encryption e;
    e.SetKeyWithIV(key, key.size(), iv);
    string res;
    StringSource s(message, true,
        new StreamTransformationFilter(e,
            new StringSink(res)
        )
    );
};
```

```

        return res;
    }

    string Decryption(SecByteBlock key, string message) {
        CBC_Mode< AES >::Decryption d;
        d.SetKeyWithIV(key, key.size(), iv);
        string res;
        StringSource s(message, true,
            new StreamTransformationFilter(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();
        cout << endl;
    }

    vector<string> FirstPass() {
        string RA = to_string(Random());
        string res = "Alice ^ " + RA;
        return { RA, res };
    }

    vector<string> SecondPass(string mess, SecByteBlock EB) {
        time_t t;
        t = time(NULL);
    }

```



```

    string TB = to_string(t);
    mess = mess + " ^ " + TB;
    string RB = to_string(Random());
    string res = "Bob ^ " + RB + " ^ " + Encryption(EB, mess);
    return { TB, RB, res };
}

```

```

vector <string> ThirdPass(string mess, SecByteBlock EB, SecByteBlock EA) {
    SecByteBlock key(AES::DEFAULT_KEYLENGTH);
    prng.GenerateBlock(key, key.size());
    string K((const char*)key.data(), key.size());
    vector <string> umess = UnMess(mess, 2);
    string nameBob = umess[0];
    string RB = umess[1];
    vector <string> umessEnc = UnMess(Decryption(EB, umess[2]), 2);
    string nameAlice = umessEnc[0];
    string RA = umessEnc[1];
    string TB = umessEnc[2];
    string res1 = Encryption(EA, nameBob + " ^ " + RA + " ^ " + K + " ^ " +
TB);
    string res2 = Encryption(EB, nameAlice + " ^ " + K + " ^ " + TB);
    return { res1 , res2, RB, K };
}

```

```

vector <string> FourthPass(vector <string> mess, SecByteBlock EA) {
    vector <string> umess = UnMess(Decryption(EA, mess[0]), 3);
    string RA = umess[1];
    string k = umess[2];
    SecByteBlock K((const byte*)k.data(), k.size());
    string res = Encryption(K, mess[2]);
    return { mess[1], res, k, RA };
}

```

```

vector <string> FifthPass(string mess1, string mess2, SecByteBlock EB) {
    vector <string> umess = UnMess(Decryption(EB, mess1), 2);
    string k = umess[1];
    string TB = umess[2];
    SecByteBlock K((const byte*)k.data(), k.size());
    string RB = Decryption(K, mess2);
    return { k, TB, RB };
}

```

```

vector <string> Check1Pass(string mess) {
    string RsA = to_string(Random());
    return { RsA, mess };
}

```

```

vector <string> Check2Pass(string RsA, string k) {
    string RsB = to_string(Random());
    SecByteBlock K((const byte*)k.data(), k.size());
    return { RsB, Encryption(K, RsA)};
}

```

```
}
```

```
vector <string> Check3Pass(string k, string RsB, string mess) {  
    SecByteBlock K((const byte*)k.data(), k.size());  
    return { Decryption(K, mess), Encryption(K, RsB) };  
}
```

```
string Check4Pass(string k, string mess) {  
    SecByteBlock K((const byte*)k.data(), k.size());  
    return Decryption(K, mess);  
}
```

```
void NeumanStubblebine(int l) {  
    prng.GenerateBlock(iv, iv.size());  
    pair <cpp_int, cpp_int> minMax = MinMax(l);  
    minim = minMax.first;  
    maxim = minMax.second;  
  
    //1й проход  
    vector <string> resFirstPass = FirstPass();  
    cout << "\nАлиса генерирует случайное число Ra = ";  
    PrintRes(resFirstPass[0]);  
    cout << "И отправляет Бобу сообщение (A, Ra) = ";  
    PrintRes(resFirstPass[1]);  
  
    //генерация ключей трентом  
    SecByteBlock EA(AES::DEFAULT_KEYLENGTH);  
    prng.GenerateBlock(EA, EA.size());  
    SecByteBlock EB(AES::DEFAULT_KEYLENGTH);  
    prng.GenerateBlock(EB, EB.size());  
  
    //2й проход  
    vector <string> resSecondPass = SecondPass(resFirstPass[1], EB);  
    cout << "\nМетка времени Tb = ";  
    PrintRes(resSecondPass[0]);  
    cout << "Боб генерирует случайное число Rb = ";  
    PrintRes(resSecondPass[1]);  
    cout << "И отправляет Тренту сообщение (B, Rb, Eb(A, Ra, Tb))\n ";  
    PrintRes(resSecondPass[2]);  
  
    //3й проход  
    vector <string> resThirdPass = ThirdPass(resSecondPass[2], EB, EA);  
    cout << "\nТрент генерирует сеансовый ключ K = ";  
    PrintRes(resThirdPass[3]);  
    cout << "И посылает Алисе следующие сообщения:\n";  
    cout << "    Ea(B, Ra, K, Tb) = ";  
    PrintRes(resThirdPass[0]);  
    cout << "    Eb(A, K, Tb) = ";  
    PrintRes(resThirdPass[1]);  
    cout << "    Rb = ";  
    PrintRes(resThirdPass[2]);  
  
    //4й проход
```

```

vector <string> fourthPass = FourthPass(resThirdPass, EA);
cout << "\nАлиса извлекает из полученного сообщения:\n";
cout << "    ключ K = ";
PrintRes(fourthPass[2]);
cout << "    своё секретное число Ra = ";
PrintRes(fourthPass[3]);
if (resFirstPass[0] == fourthPass[3])
    cout << "    (Случайное число Алисы полученное из сообщения и
полученное на первом проходе совпадает)\n";
else
    cout << "    (Случайное число Алисы полученное из сообщения и
полученное на первом проходе не совпадает)\n";
cout << "Алиса посылает Бобу следующие сообщения:\n";
cout << "    Eb(A, K, Tb) = ";
PrintRes(fourthPass[0]);
cout << "    Ek(Rb) = ";
PrintRes(fourthPass[1]);

//5й проход
vector <string> fifthPass = FifthPass(fourthPass[0], fourthPass[1], EB);
cout << "\nБоб извлекает из полученных сообщений:\n";
cout << "    ключ K = ";
PrintRes(fifthPass[0]);
cout << "    метку времени Tb = ";
PrintRes(fifthPass[1]);
cout << "    своё секретное число Rb = ";
PrintRes(fifthPass[2]);
if (fifthPass[1] == resSecondPass[0])
    cout << "    (Временная метка полученная из сообщения и полученная на
втором проходе совпадает)\n";
else
    cout << "    (Временная метка полученная из сообщения и полученная на
втором проходе не совпадает)\n";
if (fifthPass[2] == resSecondPass[1])
    cout << "    (Случайное число Боба полученное из сообщения и
полученное на втором проходе совпадает)\n";
else
    cout << "    (Случайное число Боба полученное из сообщения и
полученное на втором проходе не совпадает)\n";
if (fifthPass[0] == fourthPass[2])
    cout << "\nКлючи, полученные Алисой и Бобом, совпали\n";
else
    cout << "\nКлючи, полученные Алисой и Бобом, не совпали\n";

//повторная проверка
cout << "\n\nПовторная проверка:\n";
//1
vector <string> resCheck1Pass = Check1Pass(resThirdPass[1]);
cout << "Алиса отправляет Бобу сообщения:\n";
cout << "    Eb(A, K, Tb) = ";
PrintRes(resCheck1Pass[1]);
cout << "    R'a = ";
PrintRes(resCheck1Pass[0]);

//2

```

```

        vector<string> resCheck2Pass = Check2Pass(resCheck1Pass[0],
fifthPass[0]);
        cout << "\nБоб отправляет Алисе сообщения:\n";
        cout << "    R'b = ";
        PrintRes(resCheck2Pass[0]);
        cout << "    Ek(R'a) = ";
        PrintRes(resCheck2Pass[1]);

        //3
        vector<string> resCheck3Pass = Check3Pass(fourthPass[2],
resCheck2Pass[0], resCheck2Pass[1]);
        cout << "\nАлиса извлекает:\n    R'a = ";
        PrintRes(resCheck3Pass[0]);
        cout << "И отправляет Бобу сообщение:\n";
        cout << "    Ek(R'b) = ";
        PrintRes(resCheck3Pass[1]);

        //4
        string resCheck4Pass = Check4Pass(fifthPass[0], resCheck3Pass[1]);
        cout << "\nБоб извлекает:\n    R'b = ";
        PrintRes(resCheck4Pass);
        if (resCheck3Pass[0] == resCheck1Pass[0])
            cout << "\nПолученное и отправленное R'a совпадает\n";
        else
            cout << "\nПолученное и отправленное R'a не совпадает\n";
        if (resCheck4Pass == resCheck2Pass[0])
            cout << "Полученное и отправленное R'b совпадает\n";
        else
            cout << "Полученное и отправленное R'b не совпадает\n";
    }

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "Russian");
    int l;
    if (argc == 1) {
        cerr << "Необходимо ввести l - битовую длину случайных чисел Алисы и
Боба\n";
        return 0;
    }
    try {
        l = stoi(argv[1]);
    }
    catch (std::invalid_argument) {
        cerr << "Число l должно быть целым\n";
        return 0;
    }
    if (l < 2) {
        cerr << "Число l должно быть больше 1\n";
        return 0;
    }
    cout << "Протокол Neuman-Stubblebine\n";
    NeumanStubblebine(l);
    return 0;
}

```