

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
Яхина Шамяля Илдусовича

Преподаватель

аспирант

Р. А. Фарахутдинов

подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Теоретические сведения	4
1.1 Описание алгоритма протокола «станция-станция»	4
2 Практическая реализация	6
2.1 Описание программы	6
2.2 Тестирование программы.....	6

ВВЕДЕНИЕ

Цель работы - реализация протокола обмена ключами Ньюмана-Стаблбайна.

1 Теоретические сведения

Протокол Ньюмана-Стаблбайна является усовершенствованной версией протокола Yahalom. Его особенностью является отсутствие необходимости синхронизации часов у сторон, а также возможность повторной аутентификации без использования промежуточной стороны.

При десинхронизации часов большинство протоколов, использующих метку времени и время жизни (lifespan) сеансового ключа могут быть вскрыты. Если часы отправителя опережают часы получателя, Мэллори может перехватить сообщение отправителя и передать его повторно, когда на узле получателя метка времени сравнивается с текущей. Такая атака называется атакой повторной передачи или повторного воспроизведения. Протокол Ньюмана-Стаблбайна противодействует этой атаке.

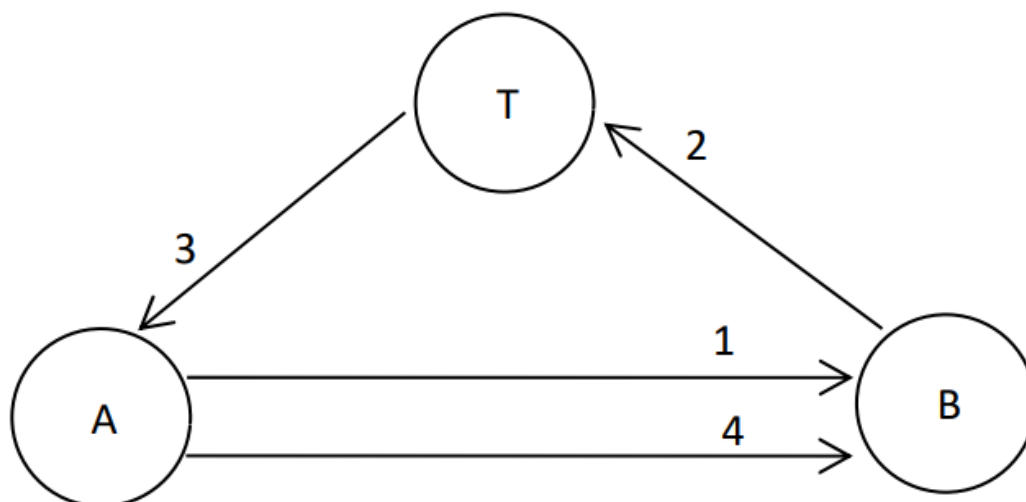


Рисунок 1 – Схема протокола Ньюмана-Стаблбайна

1.1 Описание алгоритма протокола «станция-станция»

Алгоритм протокола «станция-станция».

Вход: Целое число p_l , где p_l - битовая длина модуля случайных чисел R_A и R_B , генерируемых Алисой и Бобом.

Выход: Секретный ключ K .

1. Алиса передает Бобу $\{A, R_A\}$, где A - имя Алисы, R_A - случайное число Алисы;
2. Боб передает Тренту $\{B, R_B, E_{BT}(A, R_A, T_B)\}$, где B - имя Боба, R_B - случайное число Боба, T_B - метка времени Боба;

3. Трент передает Алисе $\{E_{AT}(B, R_A, K_{AB}, T_B), E_{BT}(A, K_{AB}, T_B), R_B\}$;
4. Алиса расшифровывает первое сообщение $D_{AT}(\{E_{AT}(B, R_A, K_{AB}, T_B)\}) = \{B, R_A, K_{AB}, T_B\}$, если R_A совпадает со значением, посланным на этапе 1, то протокол продолжается;
5. Алиса передает Бобу $\{E_{BT}(A, K_{AB}, T_B), E_{AB}(R_B)\}$, где $E_{AB}(R_B)$ - это R_B , зашифрованное ключом K_{AB} ;
6. Боб расшифровывает последовательно оба сообщения соответствующими ключами: $D_{BT}(E_{BT}(\{A, K_{AB}, T_B\})) = \{A, K_{AB}, T_B\}$ и $D_{AB}(E_{AB}(R_B)) = R_B$. Если T_B и R_B совпадают со значениями, посланными на этапе 2, то протокол заканчивается.

Далее, Алиса и Боб используют K_{AB} для своего сеанса связи.

И так как метка времени устанавливается только по часам Боба, и только Боб проверяет собственную метку времени, синхронизация часов не нужна. Протокол имеет возможность в течение заранее заданного интервала времени после его проведения Алисе и Бобу повторно проверить подлинность друг друга (провести повторную аутентификацию) без обращения к Тренту, проведя следующий трёхпроходный протокол с новыми случайными числами.

1. Алиса передает Бобу $\{E_{BT}(A, R_A, T_B), R'_A\}$, т.е. одно из сообщений Трента из прохода 3 протокола Ньюмана-Стаблбайна, а также случайное число R'_A ;
2. Боб передает Алисе $\{R'_B, E_{AB}(R'_A)\}$. Сверяя расшифрованное $E_{AB}(R'_A)$ с отправленным R'_A на первом проходе, Алиса убеждается в подлинности Боба;
3. Алиса передает Бобу $\{E_{AB}(R'_B)\}$. Сверяя расшифрованное $E_{AB}(R'_B)$ с отправленным R'_B на втором проходе, Боб убеждается в подлинности Алисы. Новые случайные числа предотвращают атаку с повторной передачей.

2 Практическая реализация

2.1 Описание программы

Все шаги алгоритма происходят в функции *NeumanStubblebineVerification*. Для генерации ключей Алисы и Боба для AES-шифрования используется функция *generateRandomHexKey*.

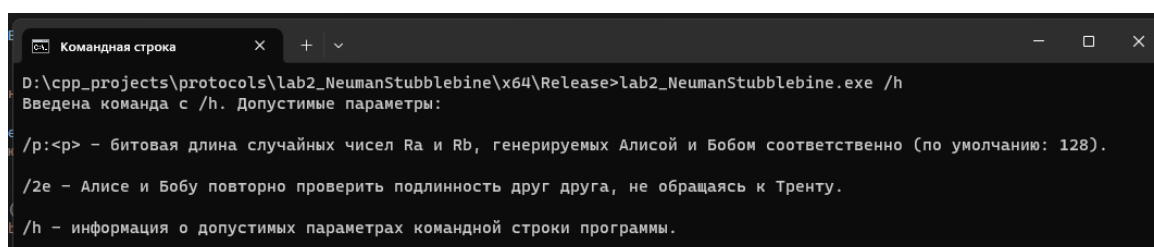
Функция *getTimeStamp* возвращает строку с меткой времени в формате "2023-10-10 18:26:03".

AES-шифрование реализовано в функции *encryptAES*, а дешифрование в функции *decryptAES*.

Описание дополнительных функций, используемых в программе: *byteArray ToHexString* - функция преобразования массива байтов в строку в шестнадцатеричном формате, *splitString* - разделение строки в вектор строк по выбранному символу, *rand_large_by_bit_length* - генерация случайного числа в выбранном промежутке.

2.2 Тестирование программы

На рисунке 2 показан вызов параметра *help*, который выводит информацию о допустимых параметрах командной строки программы.



```
Командная строка
D:\cpp_projects\protocols\lab2_NeumanStubblebine\x64\Release>lab2_NeumanStubblebine.exe /h
Введена команда с /h. Допустимые параметры:

/p:<p> - битовая длина случайных чисел Ra и Rb, генерируемых Алисой и Бобом соответственно (по умолчанию: 128).

/2e - Алисе и Бобу повторно проверить подлинность друг друга, не обращаясь к Тренту.

/h - информация о допустимых параметрах командной строки программы.
```

Рисунок 2 – Вызов параметра *help*

В примере, показанном на рисунках 3 и 4, задается параметр $p_l = 256$.

```
Командная строка
D:\cpp_projects\protocols\lab2_NeumanStubblebine\x64\Release>lab2_NeumanStubblebine.exe /p:256

p = 256

Демонстрация работы протокола "Ньюмана-Стаблбайна"

Bob & Trent AES key: AEF26C99HA54DA1EC70BC2C1C23B528
Alice & Trent AES key: 595BEE708C1F83C27E6E983707B4613D

[Этап 1] АЛИСА:

> Сгенерировала случайное число Ra = 36107374818869439472285497332614531574288742228239084264831086673089814286004
> Объединила свое имя с числом Ra: Alice,36107374818869439472285497332614531574288742228239084264831086673089814286004 и отправила данное сообщение Бобу

[Этап 2] БОБ:

> Получил сообщение Алисы : Alice,36107374818869439472285497332614531574288742228239084264831086673089814286004
> Узнал:
  имя Алисы A: Alice;
  случайное число Алисы Ra: 36107374818869439472285497332614531574288742228239084264831086673089814286004
> Объединил имя Алисы, ее случайное число и метку времени: Alice,36107374818869439472285497332614531574288742228239084264831086673089814286004,2023-12-12 19:13:21
> Зашифровал данное сообщение ключом (Bob & Trent): f0518b6dffc6697302b49ab06fd221880e3853f219e5a0f3b717453e74e61a65fb45bd62c9a45fae6e37dab9c0e34e1b4d0abdd59f3e27166a6c69ec914695ce8a07c6803c2dbf9451211e9d1e8cab6aaa71f556d97bce69d020380c38d07f5680ab48bf2166545e9525875d5b35b
> Сгенерировал случайное число Rb = 15524084818099942981410578679171497390148729163540363486694457728826825714989
> Объединил свое имя с числом Rb и зашифрованным сообщением и отправил данное сообщение Тренту

[Этап 3] ТРЕНТ:

> Получил сообщение Боба
> Узнал:
  имя Боба B: Bob;
  случайное число Боба Rb: 15524084818099942981410578679171497390148729163540363486694457728826825714989;
  имя Алисы A: Alice;
  случайное число Алисы Ra: 36107374818869439472285497332614531574288742228239084264831086673089814286004;
  временную метку Боба Tb: 2023-12-12 19:13:21
> Сгенерировал случайный сессионный ключ K = 10E256E80D088072A0A7A60267221B5
> Создал первое сообщение (B, Ra, K, Tb): Bob,36107374818869439472285497332614531574288742228239084264831086673089814286004,10E256E80D088072A0A7A60267221B5,2023-12-12 19:13:21
> Зашифровал первое сообщение ключом (Alice & Trent): 38dd666b327296fae314bed90bdafe0f3be13648cb1903a6aa536bfc4737037b94484c668cdabf724b08f46fc88de07bf788b5a078955f64681370119cfb193bd28eb9a57f9ebf79501bb08f31a48c87124199a15f3174c698324bce86a4c98c73d8ce910a6b620215dd45e8f83c7fffb4bd576c11de4af15f2f92e68ef8d2e5e8906973cd4b9024265ccdf1fed3323219
> Создал второе сообщение (A, K, Tb): Alice,10E256E80D088072A0A7A60267221B5,2023-12-12 19:13:21
> Зашифровал второе сообщение ключом (Bob & Trent): d87b48438011b6053e828f21c9f40cfcfb36b59569de2eb3b2e62ce4c5bde69cb55d38ee19f1bb45d827013286d5f3c6da0a76c0843afc6f710620d61dd7f021
> Отправил Алисе два сообщения и Rb
```

Рисунок 3 – Первый пример корректной работы протокола Ньюмана-Стаблбайна

```
Командная строка

[Этап 4] АЛИСА:

> Получила сообщения Трента:
  Первое сообщение Трента: 38dd666b327296fae314bed90bdafe0f3be13648cb1903a6aa536bfc4737037b94484c668cdabf724b08f46fc88de07bf788b5a078955f64681370119cfb193bd28eb9a57f9ebf79501bb08f31a48c87124199a15f3174c698324bce86a4c98c73d8ce910a6b620215dd45e8f83c7fffb4bd576c11de4af15f2f92e68ef8d2e5e8906973cd4b9024265ccdf1fed3323219
  Второе сообщение Трента: d87b48438011b6053e828f21c9f40cfcfb36b59569de2eb3b2e62ce4c5bde69cb55d38ee19f1bb45d827013286d5f3c6da0a76c0843afc6f710620d61dd7f021
  Случайное число Боба Rb: 15524084818099942981410578679171497390148729163540363486694457728826825714989
> Расшифровала первое сообщение Трента и узнала:
  Имя Боба B: Bob
  Случайное число Алисы Ra: 36107374818869439472285497332614531574288742228239084264831086673089814286004
  Сеансовый ключ K: 10E256E80D088072A0A7A60267221B5
  Временную метку Боба Tb: 2023-12-12 19:13:21
> Убедилась, что полученное Ra совпадает с Ra, отправленным на первом этапе
> Зашифровала Rb сеансовым ключом K и получила EK: c9025728558a478485bd3083d5815e11bfdb68dee26632b35ff87706219cb6ee3b058f4e728f54d5d4c92c1fc8555a5a658d823b12fcc36a7479717e06b7ba10cda05
> Отправила Бобу второе сообщение Трента и EK

[Этап 5] БОБ:

> Получил сообщения Алисы:
  Первое сообщение Алисы (Второе сообщение Трента): d87b48438011b6053e828f21c9f40cfcfb36b59569de2eb3b2e62ce4c5bde69cb55d38ee19f1bb45d827013286d5f3c6da0a76c0843afc6f710620d61dd7f021
  Второе сообщение Алисы: c9025728558a478485bd3083d5815e11bfdb68dee26632b35ff87706219cb6ee3b058f4e728f54d5d4c92c1fc8555a5a658d823b12fcc36a7479717e06b7ba10cda05
> Расшифровал первое сообщение Алисы (Второе сообщение Трента) и узнал:
  Имя Алисы A: Alice
  Сеансовый ключ K: 10E256E80D088072A0A7A60267221B5
  Временную метку Боба Tb: 2023-12-12 19:13:21
> Убедился, что полученная временная метка t_b совпадает с t_b, отправленной на втором этапе
> Расшифровал второе сообщение Алисы сеансовым ключом K и получил Rb:15524084818099942981410578679171497390148729163540363486694457728826825714989
> Убедился, что полученное Rb совпадает с Rb, отправленным на втором этапе

АЛИСА И БОБ УБЕДИЛИСЬ В ПОДЛИННОСТИ ДРУГ ДРУГА И ПОЛУЧИЛИ СЕКРЕТНЫЙ КЛЮЧ K = 10E256E80D088072A0A7A60267221B5
```

Рисунок 4 – Первый пример корректной работы протокола Ньюмана-Стаблбайна

В примере, показанном на рисунках 5 и 6, задаются параметры $p_l = 256$ и $2e$, чтобы запустить проверку подлинности без обращения к третьей стороне.

```
Командная строка
D:\cpp_projects\protocols\lab2_NewmanStubblebine\x64\Release>lab2_NewmanStubblebine.exe /p:256 /2e
p = 256

Демонстрация работы протокола "Ньюмана-Стаблблайна"

Bob & Trent AES key: A8H3F06FEDCE43H3199A559ABFF4225
Alice & Trent AES key: 88424FC08CB61D50DF1668586F3E8AAD

[Этап 1] АЛИСА:
> Сгенерировала случайное число Ra = 986771916999621644880688492700106010762916997679469301853404482597666507232
> Объединила свое имя с числом Ra: Alice,986771916999621644880688492700106010762916997679469301853404482597666507232 и отправила данное сообщение Бобу

[Этап 2] БОБ:
> Получил сообщение Алисы : Alice,986771916999621644880688492700106010762916997679469301853404482597666507232
> Узнал:
  имя Алисы A: Alice;
  случайное число Алисы Ra: 986771916999621644880688492700106010762916997679469301853404482597666507232
> Объединил имя Алисы, ее случайное число и метку времени: Alice,986771916999621644880688492700106010762916997679469301853404482597666507232, 2023-12-12 19:15:48
> Зашифровал данное сообщение ключом (Bob & Trent): 1df4f5he238eueu43df409dd6-23d3f7856c8a3d7d57fbb2c38c534441dc471dc122803a8227c5720ca95b6426ba20b62a3701c08413aa5d18f0d24
1a466092272150ed05cd35b64cd47e87fed77aa9f8232a20d09309eeefc6a66d7a3990970e08dec11b3b31c-f70074c56dc
> Сгенерировал случайное число Rb = 4735510169519909312194201198823720158874661271220201397020660704505010240053
> Объединил свое имя с числом Rb и зашифрованным сообщением и отправил данное сообщение Тренту

[Этап 3] ТРЕНТ:
> Получил сообщение Боба
> Узнал:
  имя Боба B: Bob;
  случайное число Боба Rb: 4735510169519909312194201198823720158874661271220201397020660704505010240053;
  имя Алисы A: Alice;
  случайное число Алисы Ra: 986771916999621644880688492700106010762916997679469301853404482597666507232;
  временную метку Боба Tb: 2023-12-12 19:15:48
> Сгенерировал случайный сессионный ключ K = 468B77A0D0A3D1AE045318D19841E542
> Создал первое сообщение (B, Ra, K, Tb): Bob,986771916999621644880688492700106010762916997679469301853404482597666507232, 468B77A0D0A3D1AE045318D19841E542, 2023-12-12 19:15:48
> Зашифровал первое сообщение ключом (Alice & Trent): ba6a15ed102db8d3e79fdac78bd4dea39ae7fd3b5712ad2ed8fe7c483b45cbda86ab6ab90e8dd75e0605512a5732a3c563b06a8058e5e1dd44f17274dda7335bd82a3ab8f7b6158d
74dda7335bd82a3ab8f7b6158db6bcb7e5c9c980a77ff92a2390f7b71e9ad265ce86e8b7626c8c30fd4aa3f13341dd28c0cc37982a91d096f390af0750c988df65d4bcc4730bc614f71a344a267889fd47a5ca757
> Создал второе сообщение (A, K, Tb): Alice,468B77A0D0A3D1AE045318D19841E542, 2023-12-12 19:15:48
> Зашифровал второе сообщение ключом (Bob & Trent): e67b01ffdaf679ffc9d9576d2c30509665cab62c0163eaa6debi1cc29b160ad098e4599fddf041c69c16de55174bde1f0f50ea72fa7f0f3bb639f11c
837ad95
> Отправил Алисе два сообщения и Rb
```

Рисунок 5 – Второй пример работы протокола Ньюмана-Стаблблайна

```
Командная строка

[Этап 4] АЛИСА:
> Получила сообщения Трента:
  Первое сообщение Трента: ba6a15ed102db8d3e79fdac78bd4dea39ae7fd3b5712ad2ed8fe7c483b45cbda86ab6ab90e8dd75e0605512a5732a3c563b06a8058e5e1dd44f17274dda7335bd82a3ab8f7b6158d
b6bcb7e5c9c980a77ff92a2390f7b71e9ad265ce86e8b7626c8c30fd4aa3f13341dd28c0cc37982a91d096f390af0750c988df65d4bcc4730bc614f71a344a267889fd47a5ca757
  Второе сообщение Трента: e67b01ffdaf679ffc9d9576d2c30509665cab62c0163eaa6debi1cc29b160ad098e4599fddf041c69c16de55174bde1f0f50ea72fa7f0f3bb639f11c837ad95
  Случайное число Боба Rb: 4735510169519909312194201198823720158874661271220201397020660704505010240053
> Расшифровала первое сообщение Трента и узнала:
  имя Боба B: Bob
  Случайное число Алисы Ra: 986771916999621644880688492700106010762916997679469301853404482597666507232
  Сеансовый ключ K: 468B77A0D0A3D1AE045318D19841E542
  Временную метку Боба Tb: 2023-12-12 19:15:48
> Убедилась, что полученное Ra совпадает с Ra, отправленным на первом этапе
> Зашифровала Rb сеансовым ключом K и получила Ek: 6ddd821360787dabfe1922f0b09c2ec2ec6d2b311f9d24c6c7886a2bd4fd610df675534bf76f36c21967b19a0bb766b9e16dd1d470912b127e74cd57
014036608f9905cd46cb16d092db7d8ff0cc3d
> Отправила Бобу второе сообщение Трента и Ek

[Этап 5] БОБ:
> Получил сообщения Алисы:
  Первое сообщение Алисы (Второе сообщение Трента): e67b01ffdaf679ffc9d9576d2c30509665cab62c0163eaa6debi1cc29b160ad098e4599fddf041c69c16de55174bde1f0f50ea72fa7f0f3bb639f11
c837ad95
  Второе сообщение Алисы: 6ddd821360787dabfe1922f0b09c2ec2ec6d2b311f9d24c6c7886a2bd4fd610df675534bf76f36c21967b19a0bb766b9e16dd1d470912b127e74cd57014636608f9905cd46cb16d
d092db7d8ff0cc3d
> Расшифровал первое сообщение Алисы (Второе сообщение Трента) и узнал:
  имя Алисы A: Alice
  Сеансовый ключ K: 468B77A0D0A3D1AE045318D19841E542
  Временную метку Боба Tb: 2023-12-12 19:15:48
> Убедился, что полученная временная метка t_b совпадает с t_b, отправленной на втором этапе
> Расшифровал второе сообщение Алисы сеансовым ключом K и получил Rb: 4735510169519909312194201198823720158874661271220201397020660704505010240053
> Убедился, что полученное Rb совпадает с Rb, отправленным на втором этапе

АЛИСА И БОБ УБЕДИЛИСЬ В ПОДЛИННОСТИ ДРУГ ДРУГА И ПОЛУЧИЛИ СЕКРЕТНЫЙ КЛЮЧ K = 468B77A0D0A3D1AE045318D19841E542
```

Рисунок 6 – Второй пример работы протокола Ньюмана-Стаблблайна

ПРИЛОЖЕНИЕ А

Листинг программы

```
#include <iostream>
#include <vector>
#include <chrono>
#include <time.h>
#include <boost/random/random_device.hpp>
#include <boost/multiprecision/cpp_int.hpp>
#include <boost/random.hpp>
#include <sstream>
#include <fstream>
#include <unordered_map>
#include <string>
#include <windows.h>
#include <cryptlib.h>
#include "rijndael.h"
#include "modes.h"
#include "files.h"
#include "osrng.h"
#include "hex.h"
#include <unordered_set>
#include <ctime>

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

const int AES_KEY_SIZE = AES::DEFAULT_KEYLENGTH;
const int AES_BLOCK_SIZE = AES::BLOCKSIZE;

string byteArrayToHexString(const byte* input, size_t length) {
    ostringstream ss;
    ss << hex << setfill('0');
    for (size_t i = 0; i < length; ++i)
        ss << setw(2) << static_cast<int>(input[i]);
    return ss.str();
}

string encryptAES(const string& plainText, const string& hexKey) {
    SecByteBlock key((const byte*)hexKey.data(), AES_BLOCK_SIZE);
    ECB_Mode<AES>::Encryption encryptor;
    encryptor.SetKey(key, key.size());
    string cipherText;
    StringSource(plainText, true, new StreamTransformationFilter(encryptor, new
StringSink(cipherText)));
    return cipherText;
}

string decryptAES(const string& cipherText, const string& hexKey) {
    SecByteBlock key((const byte*)hexKey.data(), AES_BLOCK_SIZE);
    ECB_Mode<AES>::Decryption decryptor;
    decryptor.SetKey(key, key.size());
    string decryptedText;
    StringSource(cipherText, true, new StreamTransformationFilter(decryptor, new
StringSink(decryptedText)));
    return decryptedText;
}
```

```

string generateRandomHexKey(int keySize) {
    AutoSeededRandomPool prng;
    SecByteBlock key(keySize);
    prng.GenerateBlock(key, keySize);
    string hexKey;
    HexEncoder encoder(new StringSink(hexKey));
    encoder.Put(key, keySize);
    encoder.MessageEnd();
    return hexKey;
}

string findInStr(string const& str, int n) {
    if (str.length() < n)
        return str;
    return str.substr(0, n);
}

string getTimestamp() {
    time_t currentTime = time(nullptr);
    tm timeInfo = {};
    localtime_s(&timeInfo, &currentTime);
    char buffer[80];
    strftime(buffer, 80, "%Y-%m-%d %H:%M:%S", &timeInfo);
    string timestamp(buffer);
    return timestamp;
}

vector<string> splitString(const string& input, char zn) {
    istringstream stream(input);
    string str1;
    vector<string> strs;
    while (getline(stream, str1, zn)) {
        strs.push_back(str1);
    }
    return strs;
}

cpp_int rand_large_by_bit_length(int bit_length) {
    random_device gen;
    uniform_int_distribution<int> ui(0, 1);
    cpp_int result = 0;
    for (int i = 0; i < bit_length; ++i) {
        result <= 1;
        result |= ui(gen);
    }
    return result;
}

void helpFunc() {
    cout << "Введена команда с /h. Допустимые параметры:";
    cout << "\n\n/p:<p> - битовая длина случайных чисел Ra и Rb, генерируемых Алисой и Бобом  
соответственно (по умолчанию: 128).";
    cout << "\n\n/2e - Алисе и Бобу повторно проверить подлинность друг друга, не обращаясь  
к Тренту.";
    cout << "\n\n/h - информация о допустимых параметрах командной строки программы.\n";
}

bool NeumanStubblebineVerification(int p, bool oneMoreVerification) {
    char zn = ',';
    cout << "\nДемонстрация работы протокола \"Ньюмана-Стабблбайна\"\n\n";

    //Ключи:
    const string BobTrentHexKey = generateRandomHexKey(AES_KEY_SIZE);
    cout << "\nBob & Trent AES key: " << BobTrentHexKey;
}

```

```

const string AliceTrentHexKey = generateRandomHexKey(AES_KEY_SIZE);
cout << "\nAlice & Trent AES key: " << AliceTrentHexKey;

//АЛИСА

cout << "\n\n[Этап 1] АЛИСА:\n";

cpp_int r_a = rand_large_by_bit_length(p);
string name_a = "Alice";
string str_r_a = boost::lexical_cast<string>(r_a);
cout << "\n > Сгенерировала случайное число Ra = " << str_r_a;
string alice_first_str = name_a + "," + str_r_a;
cout << "\n > Объединила свое имя с числом Ra: " << alice_first_str << " и отправила
данное сообщение Бобу";

//БОБ

cout << "\n\n[Этап 2] БОБ:\n";

cout << "\n > Получил сообщение Алисы : " << alice_first_str;
vector<string> alice_first_str_vec = splitString(alice_first_str, zn);
string BZ_name_a = alice_first_str_vec[0];
string BZ_r_a = alice_first_str_vec[1];
cout << "\n > Узнал:\n      имя Алисы A: " << BZ_name_a << "; \n      случайное число Алисы
Ra: " << BZ_r_a;
string t_b = getTimestamp();
string bob_first_str = alice_first_str + "," + t_b;
cout << "\n > Объединил имя Алисы, ее случайное число и метку времени: " <<
bob_first_str;
//2023-10-10 18:26:03
string e_b = encryptAES(bob_first_str, BobTrentHexKey);
cout << "\n > Зашифровал данное сообщение ключом (Bob & Trent): " <<
byteArrayToHexString(reinterpret_cast<const byte*>(e_b.data()), e_b.length());
string name_b = "Bob";
cpp_int r_b = rand_large_by_bit_length(p);
string str_r_b = boost::lexical_cast<string>(r_b);
cout << "\n > Сгенерировал случайное число Rb = " << str_r_b;
vector<string> bob_first_str_vec = { name_b, str_r_b, e_b };
cout << "\n > Объединил свое имя с числом Rb и зашифрованным сообщением и отправил
данное сообщение Тренту";

//ТРЕНТ

cout << "\n\n[Этап 3] ТРЕНТ:\n";

cout << "\n > Получил сообщение Боба";
string TZ_name_b = bob_first_str_vec[0];
string TZ_r_b = bob_first_str_vec[1];
string TZ_e_b = bob_first_str_vec[2];
vector<string> TZ_e_b_decrypted_vec = splitString(decryptAES(TZ_e_b, BobTrentHexKey),
zn);
string TZ_name_a = TZ_e_b_decrypted_vec[0];
string TZ_r_a = TZ_e_b_decrypted_vec[1];
string TZ_t_b = TZ_e_b_decrypted_vec[2];
cout << "\n > Узнал:\n      имя Боба B: " << TZ_name_b << "; \n      случайное число Боба Rb:
" << TZ_r_b << "; \n      имя Алисы A: " << TZ_name_a << "; \n      случайное число Алисы Ra: " <<
TZ_r_a << "; \n      временную метку Боба Tb: " << TZ_t_b;
const string SessionKey = generateRandomHexKey(AES_KEY_SIZE);
cout << "\n > Сгенерировал случайный сессионный ключ K = " << SessionKey;

string trent_first_str = TZ_name_b + "," + TZ_r_a + "," + SessionKey + "," + TZ_t_b;
cout << "\n > Создал первое сообщение (B, Ra, K, Tb): " << trent_first_str;
string e_a = encryptAES(trent_first_str, AliceTrentHexKey);

```

```

    cout << "\n > Зашифровал первое сообщение ключом (Alice & Trent): " <<
    byteArrayToHexString(reinterpret_cast<const byte*>(e_a.data()), e_a.length());

    string trent_second_str = TZ_name_a + "," + SessionKey + "," + TZ_t_b;
    cout << "\n > Создал второе сообщение (A, K, Tb): " << trent_second_str;
    e_b = encryptAES(trent_second_str, BobTrentHexKey);
    cout << "\n > Зашифровал второе сообщение ключом (Bob & Trent): " <<
    byteArrayToHexString(reinterpret_cast<const byte*>(e_b.data()), e_b.length());
    vector<string> trent_strs_vec = { e_a , e_b, TZ_r_b };
    cout << "\n > Отправил Алисе два сообщения и Rb";

    //АЛИСА

    cout << "\n\n[Этап 4] АЛИСА:\n";

    cout << "\n > Получила сообщения Трента: ";
    string AZ_trent_first_str = trent_strs_vec[0];
    cout << "\n    Первое сообщение Трента: " << byteArrayToHexString(reinterpret_cast<const
byte*>(AZ_trent_first_str.data()), AZ_trent_first_str.length());
    string AZ_trent_second_str = trent_strs_vec[1];
    cout << "\n    Второе сообщение Трента: " << byteArrayToHexString(reinterpret_cast<const
byte*>(AZ_trent_second_str.data()), AZ_trent_second_str.length());
    string AZ_r_b = trent_strs_vec[2];
    cout << "\n    Случайное число Боба Rb: " << AZ_r_b;
    vector<string> AZ_trent_first_str_decrypted_vec =
splitString(decryptAES(AZ_trent_first_str, AliceTrentHexKey), zn);
    string AZ_b_name = AZ_trent_first_str_decrypted_vec[0];
    string AZ_r_a = AZ_trent_first_str_decrypted_vec[1];
    string AZ_K = AZ_trent_first_str_decrypted_vec[2];
    string AZ_t_b = AZ_trent_first_str_decrypted_vec[3];
    cout << "\n > Расшифровала первое сообщение Трента и узнала:";
    cout << "\n    Имя Боба B: " << AZ_b_name;
    cout << "\n    Случайное число Алисы Ra: " << AZ_r_a;
    cout << "\n    Сеансовый ключ K: " << AZ_K;
    cout << "\n    Временную метку Боба Tb: " << AZ_t_b;

    if (str_r_a == AZ_r_a) {
        cout << "\n > Убедилась, что полученное Ra совпадает с Ra, отправленным на первом
этапе";
    }
    else {
        cout << "\n ERROR: Полученное Ra НЕ совпадает с Ra, отправленным на первом
этапе\n\n";
        return false;
    }

    string e_k = encryptAES(AZ_r_b, SessionKey);
    cout << "\n > Зашифровала Rb сеансовым ключом K и получила Ek: " <<
    byteArrayToHexString(reinterpret_cast<const byte*>(e_k.data()), e_k.length());
    vector<string> alice_second_str = { AZ_trent_second_str , e_k };
    cout << "\n > Отправила Бобу второе сообщение Трента и Ek";

    //БОБ

    cout << "\n\n[Этап 5] БОБ:\n";

    cout << "\n > Получил сообщения Алисы: ";
    string BZ2_trent_second_str = alice_second_str[0];
    cout << "\n    Первое сообщение Алисы (Второе сообщение Трента): " <<
    byteArrayToHexString(reinterpret_cast<const byte*>(BZ2_trent_second_str.data()),
BZ2_trent_second_str.length());
    string BZ2_e_k = alice_second_str[1];
    cout << "\n    Второе сообщение Алисы: " << byteArrayToHexString(reinterpret_cast<const
byte*>(BZ2_e_k.data()), BZ2_e_k.length());

```

```

    vector<string> BZ2_trent_second_str_decrypted_vec =
splitString(decryptAES(BZ2_trent_second_str, BobTrentHexKey), zn);
    string BZ2_a_name = BZ2_trent_second_str_decrypted_vec[0];
    string BZ2_K = BZ2_trent_second_str_decrypted_vec[1];
    string BZ2_t_b = BZ2_trent_second_str_decrypted_vec[2];
    cout << "\n > Расшифровал первое сообщение Алисы (Второе сообщение Трента) и узнал:";
    cout << "\n    Имя Алисы A: " << BZ2_a_name;
    cout << "\n    Сеансовый ключ K: " << BZ2_K;
    cout << "\n    Временную метку Боба Tb: " << BZ2_t_b;

    if (t_b == BZ2_t_b) {
        cout << "\n > Убедился, что полученная временная метка t_b совпадает с t_b,
отправленной на втором этапе";
    }
    else {
        cout << "\n ERROR: Полученное t_b НЕ совпадает с t_b, отправленным на втором
этапе\n\n";
        return false;
    }

    string BZ2_r_b = decryptAES(BZ2_e_k, BZ2_K);
    cout << "\n > Расшифровал второе сообщение Алисы сеансовым ключом K и получил Rb:" <<
BZ2_r_b;

    if (str_r_b == BZ2_r_b) {
        cout << "\n > Убедился, что полученное Rb совпадает с Rb, отправленным на втором
этапе";
    }
    else {
        cout << "\n ERROR: Полученное Rb НЕ совпадает с Rb, отправленным на втором
этапе\n\n";
        return false;
    }

    cout << "\n\n АЛИСА И БОБ УБЕДИЛИСЬ В ПОДЛИННОСТИ ДРУГ ДРУГА И ПОЛУЧИЛИ СЕКРЕТНЫЙ КЛЮЧ K
= " << SessionKey;

    cout << "\n\n\n";
    if (!oneMoreVerification)
        return true;

    cout << "Проверка Алисой и Бобом подлинности друг друга без обращения к Тренту";

    //АЛИСА

    cout << "\n\n[Этап 1] АЛИСА:\n";

    cpp_int r_a1 = rand_large_by_bit_length(p);
    string str_r_a1 = boost::lexical_cast<string>(r_a1);
    cout << "\n > Сгенерировала случайное число Ra' = " << r_a1;
    cout << "\n > Имеет второе сообщение Трента, полученное на этапе 3: " <<
byteArrayToHexString(reinterpret_cast<const byte*>(AZ_trent_second_str.data()),
AZ_trent_second_str.length());
    vector<string> alice_new_message = { AZ_trent_second_str, str_r_a1 };
    cout << "\n > Отправила Бобу второе сообщение Трента и Ra'";

    //БОБ

    cout << "\n\n[Этап 2] БОБ:\n";

    cout << "\n > Получил сообщение Алисы";
    string BZnew_trent_second_str = alice_new_message[0];
    string BZnew_r_a = alice_new_message[1];

```

```

cpp_int r_b1 = rand_large_by_bit_length(p);
string str_r_b1 = boost::lexical_cast<string>(r_b1);
cout << "\n > Сгенерировал случайное число Rb' = " << str_r_b1;
string e_k_r_a1 = encryptAES(BZnew_r_a, BZ2_K);
cout << "\n > Зашифровал Ra' сессионным ключом K: " <<
byteArrayToHexString(reinterpret_cast<const byte*>(e_k_r_a1.data()), e_k_r_a1.length());
cout << "\n > Отправил Алисе Rb' и зашифрованное Ra'";

//АЛИСА

cout << "\n\n[Этап 3] АЛИСА:\n";

cout << "\n > Получила сообщение Боба";
string AZnew_r_a1 = decryptAES(e_k_r_a1, AZ_K);
cout << "\n > Расшифровала зашифрованное Бобом сообщение и получила Ra':" << AZnew_r_a1;
if (AZnew_r_a1 == str_r_a1)
    cout << "\n > Убедилась, что полученное Ra' совпадает с Ra', отправленным на первом
этапе";
else {
    cout << "\n ERROR: Полученное Ra' НЕ совпадает с Ra', отправленным на первом
этапе\n\n";
    return false;
}
string e_k_r_b1 = encryptAES(str_r_b1, AZ_K);
cout << "\n > Зашифровала Rb' сессионным ключом K: " <<
byteArrayToHexString(reinterpret_cast<const byte*>(e_k_r_b1.data()), e_k_r_b1.length());
cout << "\n > Отправила Бобу зашифрованное Rb'";

//БОБ

cout << "\n\n[Этап 4] БОБ:\n";

cout << "\n > Получил сообщение Алисы";
string BZnew_r_b1 = decryptAES(e_k_r_b1, BZ2_K);
cout << "\n > Расшифровал зашифрованное Алисой сообщение и получил Rb':" << AZnew_r_a1;
if (BZnew_r_b1 == str_r_b1)
    cout << "\n > Убедилась, что полученное Rb' совпадает с Rb', отправленным на втором
этапе";
else {
    cout << "\n ERROR: Полученное Rb' НЕ совпадает с Rb', отправленным на втором
этапе\n\n";
    return false;
}

cout << "\n\n АЛИСА И БОБ УБЕДИЛИСЬ В ПОДЛИННОСТИ ДРУГ ДРУГА БЕЗ ПОМОЩИ ТРЕНТА";
cout << "\n\n\n";
return true;
}

int main(int argc, char* argv[]) {
    setlocale(LC_ALL, "rus");
    int p = 128;
    bool oneMoreVerification = false;
    for (int i = 0; argv[i]; i++) {
        string checkStr = string(argv[i]);
        if (findInStr(checkStr, 2) == "/h") {
            helpFunc();
            return 0;
        }
        if (checkStr.length() > 2) {
            string ifStr = findInStr(checkStr, 3);
            string subStr = checkStr.substr(3, checkStr.length());
            char symbol = ',';

```

```

        if (ifStr == "/p:") {
            p = stoi(checkStr.substr(3, checkStr.length()));
        }
        if (ifStr == "/2e") {
            oneMoreVerification = true;
        }
    }
}
cout << "\np = " << p << "\n";
NeumanStubblebineVerification(p, oneMoreVerification);
return 0;
}

```