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 передачи секретного ключа по открытому каналу**

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 трёхпроходный (трехэтапный) протокол Шамира.

**2 Теоретические сведения**

Этот изобретённый Ади Шамиром, но никогда не опубликованный протокол, позволяет Алисе и Бобу безопасно обмениваться информацией, не используя предварительного обмена ни секретными, ни открытыми ключами. Он предполагает использование коммутативного шифра относительно транспозиции ключей, то есть для которого:

где использование ключей Алисы обозначается индексом , а Боба – индексом .

Алиса хочет передать Бобу секретный ключ . Для этого Алиса применяет следующий протокол.

**Трёхпроходный протокол Шамира**

*Вход*: Битовая длина простого числа .

*Выход*: Боб получает от Алисы секретный ключ.

Шаг 1. Алиса шифрует секретный ключ своим ключом и посылает результат Бобу ;

Шаг 2. Боб шифрует своим ключом и посылает результат Алисе ;

Шаг 3. Алиса расшифровывает своим ключом и посылает результат Бобу ;

Шаг 4. Боб расшифровывает своим ключом , получая .

Замечание. Не всякая коммутативная криптосистема сохраняет свою устойчивость в этом протоколе. Одноразовые блокноты обладают свойством коммутативности и обеспечивают абсолютную безопасность, но с описанным выше протоколом они работать не будут. При использовании одноразового блокнота три шифртекста будут выглядеть следующим образом:

Ева, записав все эти три сообщения, которыми обмениваются Алиса и Боб, просто выполнить операцию над всеми этими шифртекстами и восстановит сообщение:

.

Таким образом, данный способ шифрования работать не будет.

Криптосистема RSA вполне удовлетворяет требованиям этого протокола при условии, что Алиса и Боб использует один и тот же модуль , а свои пары открытый/закрытый ключ держат секрете.

Ади Шамир и, независимо, Джим Омура, описали алгоритм шифрования, похожий на алгоритм , который будет работать с описываемым протоколом.

**Генерация ключей**

Генерируется большое безопасное простое число , которое может быть открытым для группы пользователей.

Функция Эйлера . Для генерации пары секретных ключей пользователя выбирается случайное число с условиями и . По расширенному алгоритму Евклида вычисляется . Число условно называются *открытым ключом* (или ключом шифрования), число – *закрытым ключом* (или ключом расшифрования) пользователя .

**Шифрование и расшифрование**

Для зашифрования сообщения вычисляется .

Для расшифрования сообщения вычисляется .

Данная криптосистема работает только при условии секретности обоих (открытый/закрытый) ключей, таким образом, она является ассиметричной, но не является криптосистемой с открытым ключом (с публичным ключом). В трёхпроходном протоколе Шамира эта криптосистема работает при дополнительном условии общего модуля участвующих сторон. В этих условиях Ева не может получить , не решив проблему дискретного логарифма.

**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 введённой, при запуске программы, длинны простого числа.

Функция содержит все шаги описанного выше алгоритма, при этом для генерации простого числа используется функция из библиотеки , генерация ключей , , происходит с помощью функции .

В программе используются большие числа, работать с которыми позволяет подключённая библиотека , кроме того, силами данной библиотеки осуществляется генерация секретного ключа в функции .

Для подсчёта НОД двух целых чисел, а также для поиска обратного элемента в поле используется расширенный алгоритм Евклида – функция . Кроме того, была реализована функция быстрого возведения в степень по модулю – , которая участвует в шифровании и расшифровании.

**3.2 Результаты тестирования программы**

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

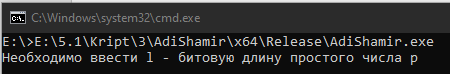


Рисунок 1 – Запуск программы без параметров

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

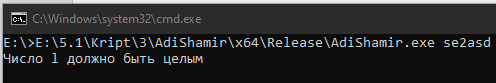


Рисунок 2 – Запуск программы со случайным набором символов в качестве параметра

Кроме того, должен быть больше – рисунок 3.

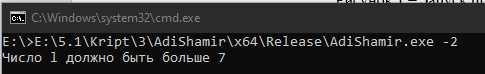


Рисунок 3 – Запуск программы с параметром меньшим семи

На рисунках 4-6 представлены успешные запуски программ.

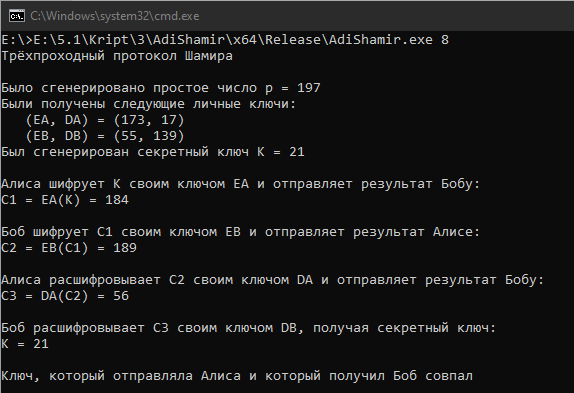


Рисунок 4 – Запуск программы с параметром

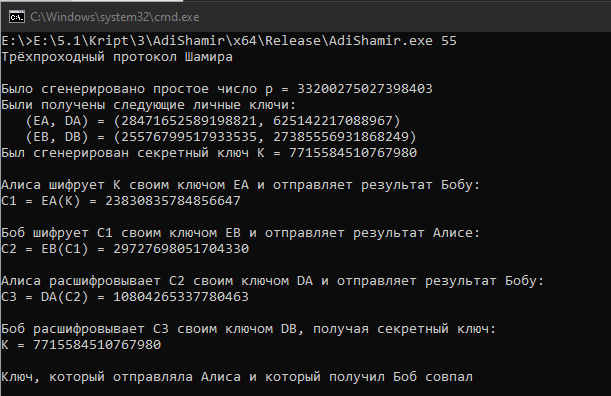


Рисунок 5 – Запуск программы с параметром

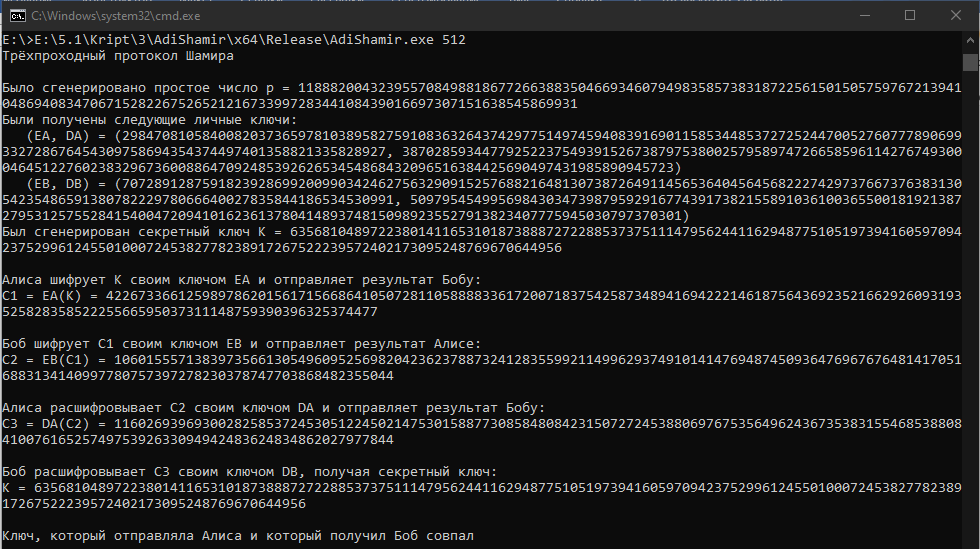


Рисунок 6 – Запуск программы с параметром

**Листинг кода**

#include <iostream>

#include <string>

#include "rsa.h"

#include "osrng.h"

#include <random>

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

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

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

using namespace std;

using namespace CryptoPP;

using namespace boost::multiprecision;

using namespace boost::random;

AutoSeededRandomPool rng;

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

while (a < 0)

a = a + p;

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;

}

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);

}

pair <cpp\_int, cpp\_int> GenKey(cpp\_int p) {

cpp\_int publicKey = Random(2, p - 2);

while (ExtendedEuclid(publicKey, p - 1)[0] != 1)

publicKey = Random(2, p - 2);

cpp\_int privateKey = ModNegative(ExtendedEuclid(publicKey, p - 1)[1], p - 1);

return make\_pair(publicKey, privateKey);

}

void AdiShamir(int l) {

//Генерация p

InvertibleRSAFunction params;

params.GenerateRandomWithKeySize(rng, l\*2);

const Integer& pstr = params.GetPrime1();

ostringstream oss;

oss << pstr;

string s(oss.str());

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

cpp\_int p(s);

cout << "\nБыло сгенерировано простое число p = " << p << "\n";

//Генерация ключей для шифрования и расшифрования

pair<cpp\_int, cpp\_int> keys = GenKey(p);

cpp\_int EA = keys.first;

cpp\_int DA = keys.second;

keys = GenKey(p);

cpp\_int EB = keys.first;

cpp\_int DB = keys.second;

cout << "Были получены следующие личные ключи:\n";

cout << " (EA, DA) = (" << EA << ", " << DA << ")\n";

cout << " (EB, DB) = (" << EB << ", " << DB << ")\n";

// Генерация секретного ключа

cpp\_int K = Random(2, p - 1);

cout << "Был сгенерирован секретный ключ K = " << K << "\n";

//1

cout << "\nАлиса шифрует K своим ключом EA и отправляет результат Бобу:\n";

cpp\_int C1 = Exponentiation(K, EA, p);

cout << "C1 = EA(K) = " << C1 << "\n";

//2

cout << "\nБоб шифрует С1 своим ключом EB и отправляет результат Алисе:\n";

cpp\_int C2 = Exponentiation(C1, EB, p);

cout << "C2 = EB(C1) = " << C2 << "\n";

//3

cout << "\nАлиса расшифровывает C2 своим ключом DA и отправляет результат Бобу:\n";

cpp\_int C3 = Exponentiation(C2, DA, p);

cout << "C3 = DA(C2) = " << C3 << "\n";

//4

cout << "\nБоб расшифровывает С3 своим ключом DB, получая секретный ключ:\n";

cpp\_int res = Exponentiation(C3, DB, p);

cout << "K = " << res << "\n";

//Проверка

if (res != K)

cout << "\nБоб получил неверное сообщение\n";

else

cout << "\nКлюч, который отправляла Алиса и который получил Боб совпал\n";

}

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

setlocale(LC\_ALL, "Russian");

int l;

if (argc == 1) {

cerr << "Необходимо ввести l - битовую длину простого числа p\n";

return 0;

}

try {

l = stoi(argv[1]);

}

catch (std::invalid\_argument) {

cerr << "Число l должно быть целым\n";

return 0;

}

if (l < 8) {

cerr << "Число l должно быть больше 7\n";

return 0;

}

cout << "Трёхпроходный протокол Шамира\n";

AdiShamir(l);

return 0;

}