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

**Схемы ЭЦП**

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 схему подписи Эль-Гамаля.

**2 Теоретические сведения**

Схема Эль-Гамаля – криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Эту схему можно использовать как для цифровых подписей, так и для шифрования.

Цифровая подпись служит для проверки изменения данных и установления подлинности подписавшейся стороны. Получатель подписанного сообщения может использовать цифровую подпись для доказательства третьей стороне того, что подпись действительно сделана отправляющей стороной.

**Схема подписи Эль-Гамаля**

*Вход*: Битовая длина простого числа .

*Выход*: Результат проверки подписи.

**Этап 1 – Генерация ключей и подписывающей стороной**

Шаг 1. Генерируется случайное простое число ;n = p ⋅ q жЖ

Шаг 2. Вычисляется целое число – первообразный корень ;

Шаг 3. Генерируется случайное число такое, что ;

Шаг 4. Вычисляется .

Открытым ключом является , закрытым ключом – число . такое, что является сообщением.

**Этап 2 – Подпись сообщения**

Шаг 1. Генерируется случайное число , взаимно простое с ;n = p ⋅ q жЖ

Шаг 2. Вычисляется ;

Шаг 3. С помощью обратного алгоритма Евклида вычисляется ;

Шаг 4. Вычисляется .

Подписью сообщения является пара , при этом значение должно храниться в секрете.

**Этап 3 – Проверка подписи**

Шаг 1. Необходимо проверить выполнение следующего условия: . Если оно выполняется, то подпись прошла проверку, иначе нет.

Каждая подпись или шифрование Эль-Гамаля требует нового значения , и это значение должно быть выбрано случайным образом. Если когда-нибудь Ева раскроет значение k, используемое Алисой, она сможет раскрыть закрытый ключ Алисы . Если Ева когда-нибудь сможет получить два сообщения, подписанные или зашифрованные с помощью одного и того же , то она сможет раскрыть , даже не зная значение .

**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 числа .

Функция содержит все шаги описанного выше алгоритма, при этом для генерации числа и нахождения первообразного корня используется функция из библиотеки .

В программе используются большие числа, работать с которыми позволяет подключённая библиотека , кроме того, силами данной библиотеки осуществляется генерация случайных чисел из заданного диапазона 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 – , часто используемая при подсчётах.

**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 четырёх

На рисунках 4-6 представлены успешные запуски программ.

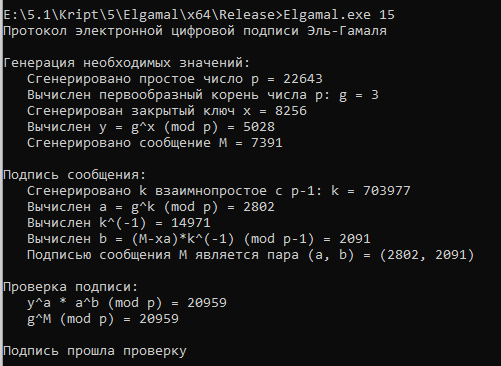


Рисунок 4 – Запуск программы с параметром

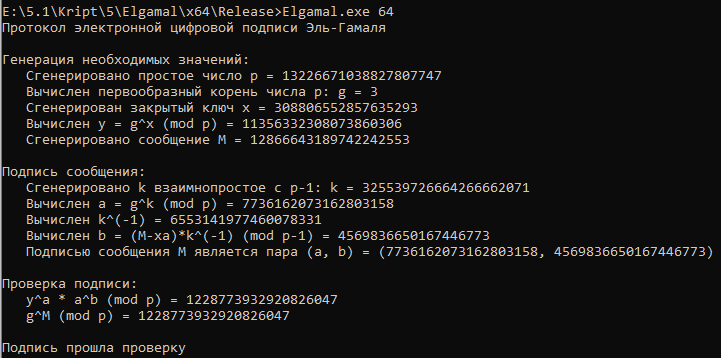


Рисунок 5 – Запуск программы с параметром

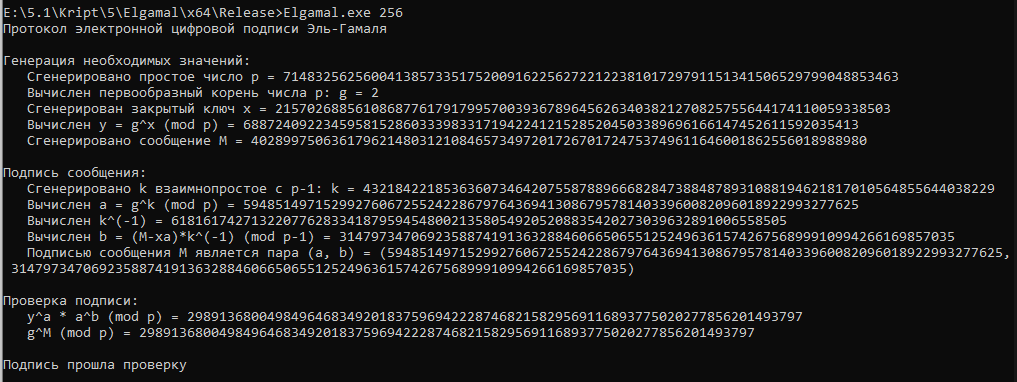


Рисунок 6 – Запуск программы с параметром

**Листинг кода**

#include <iostream>

#include <string>

#include "osrng.h"

#include "dh.h"

#include <random>

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

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

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

using namespace CryptoPP;

using namespace std;

using namespace boost::multiprecision;

using namespace boost::random;

AutoSeededRandomPool rnd;

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;

}

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;

}

void Elgamal(int l) {

DH dh;

cpp\_int p, g, x, y, M, k, a, kobr, b, left, right;

cout << "\nГенерация необходимых значений:\n";

dh.AccessGroupParameters().GenerateRandomWithKeySize(rnd, l);

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

g = IntegerToCppint(dh.GetGroupParameters().GetGenerator());

x = Random(2, p - 2);

y = Exponentiation(g, x, p);

M = Random(2, p - 2);

cout << " Cгенерировано простое число p = " << p << "\n";

cout << " Вычислен первообразный корень числа p: g = " << g << "\n";

cout << " Cгенерирован закрытый ключ x = " << x << "\n";

cout << " Вычислен y = g^x (mod p) = " << y << "\n";

cout << " Cгенерировано сообщение M = " << M << "\n";

cout << "\nПодпись сообщения:\n";

do {

k = Random(2, p \* 100);

} while (ExtendedEuclid(k, p-1)[0] != 1);

a = Exponentiation(g, k, p);

kobr = ModNegative(ExtendedEuclid(k, p - 1)[1], p - 1);

b = ModNegative(M - x \* a, p - 1) \* kobr % (p - 1);

cout << " Cгенерировано k взаимнопростое с p-1: k = " << k << "\n";

cout << " Вычислен a = g^k (mod p) = " << a << "\n";

cout << " Вычислен k^(-1) = " << kobr << "\n";

cout << " Вычислен b = (M-xa)\*k^(-1) (mod p-1) = " << b << "\n";

cout << " Подписью сообщения M является пара (a, b) = (" << a << ", " << b << ")\n";

cout << "\nПроверка подписи:\n";

left = Exponentiation(y, a, p) \* Exponentiation(a, b, p) % p;

right = Exponentiation(g, M, p);

cout << " y^a \* a^b (mod p) = " << left << "\n";

cout << " g^M (mod p) = " << right << "\n";

if (left == right)

cout << "\nПодпись прошла проверку\n";

else

cout << "\nПодпись не прошла проверку\n";

}

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

setlocale(LC\_ALL, "Russian");

int l;

if (argc == 1) {

cerr << "Необходимо ввести l - битовую длину 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 < 4) {

cerr << "Число l должно быть больше 3\n";

return 0;

}

cout << "Протокол электронной цифровой подписи Эль-Гамаля\n";

Elgamal(l);

return 0;

}