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 p.

Выход: Результат проверки подписи.

Этап 1 – Генерация ключей x и (y, g, p) подписывающей стороной

Шаг 1. Генерируется случайное простое число p;

Шаг 2. Вычисляется целое число g – первообразный корень p;

Шаг 3. Генерируется случайное число x такое, что $1 < x < p - 1$;

Шаг 4. Вычисляется $y = g^x (mod\, p)$.

Открытым ключом является (y, g, p), закрытым ключом – число x. M такое, что $0 < M < p$ является сообщением.

Этап 2 – Подпись сообщения

Шаг 1. Генерируется случайное число k, взаимно простое с $p - 1$;

Шаг 2. Вычисляется $a = g^k (mod\, p)$;

Шаг 3. С помощью обратного алгоритма Евклида вычисляется $k^{-1}\ (mod\, p - 1)$;

Шаг 4. Вычисляется $b = (m - xa) * k^{-1} (mod\, p - 1)$.

Подписью сообщения M является пара (a, b), при этом значение k должно храниться в секрете.

Этап 3 – Проверка подписи

Шаг 1. Необходимо проверить выполнение следующего условия: $y^a a^b (mod\, p) = g^M (mod\, p)$. Если оно выполняется, то подпись прошла проверку, иначе нет.

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

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 введённой, при запуске программы, длинны числа p.

Функция $Elgamal$ содержит все шаги описанного выше алгоритма, при этом для генерации числа p и нахождения первообразного корня g используется функция из библиотеки $Crypto++$.

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

Для подсчёта НОД двух целых чисел, а также для поиска обратного элемента в поле используется расширенный алгоритм Евклида – функция $ExtendedEuclid$. Кроме того, была реализована функция быстрого возведения в степень по модулю – $Exponentiation$, часто используемая при подсчётах.

3.2 Результаты тестирования программы

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

```
E:\5.1\Kript\5\Elgamal\x64\Release>Elgamal.exe
Необходимо ввести l - битовую длину числа p
```

Рисунок 1 – Запуск программы без параметров

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

```
E:\5.1\Kript\5\Elgamal\x64\Release>Elgamal.exe adsgad
Число l должно быть целым
```

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

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

```
E:\5.1\Kript\5\Elgamal\x64\Release>Elgamal.exe 1
Число l должно быть больше 3
```

Рисунок 3 – Запуск программы с параметром меньшим четырёх

На рисунках 4-6 представлены успешные запуски программ.

```
E:\5.1\Kript\5\Elgamal\x64\Release>Elgamal.exe 15
Протокол электронной цифровой подписи Эль-Гамаля

Генерация необходимых значений:
   Сгенерировано простое число p = 22643
   Вычислен первообразный корень числа p: g = 3
   Сгенерирован закрытый ключ x = 8256
   Вычислен y = g^x (mod p) = 5028
   Сгенерировано сообщение M = 7391

Подпись сообщения:
   Сгенерировано k взаимнопростое с p-1: k = 703977
   Вычислен a = g^k (mod p) = 2802
   Вычислен k^(-1) = 14971
   Вычислен b = (M-xa)*k^(-1) (mod p-1) = 2091
   Подписью сообщения M является пара (a, b) = (2802, 2091)

Проверка подписи:
   y^a * a^b (mod p) = 20959
   g^M (mod p) = 20959

Подпись прошла проверку
```

Рисунок 4 – Запуск программы с параметром $l = 15$

```
E:\5.1\Kript\5\Elgamal\x64\Release>Elgamal.exe 64
Протокол электронной цифровой подписи Эль-Гамаля

Генерация необходимых значений:
   Сгенерировано простое число p = 13226671038827807747
   Вычислен первообразный корень числа p: g = 3
   Сгенерирован закрытый ключ x = 308806552857635293
   Вычислен y = g^x (mod p) = 11356332308073860306
   Сгенерировано сообщение M = 12866643189742242553

Подпись сообщения:
   Сгенерировано k взаимнопростое с p-1: k = 325539726664266662071
   Вычислен a = g^k (mod p) = 7736162073162803158
   Вычислен k^(-1) = 6553141977460078331
   Вычислен b = (M-xa)*k^(-1) (mod p-1) = 4569836650167446773
   Подписью сообщения M является пара (a, b) = (7736162073162803158, 4569836650167446773)

Проверка подписи:
   y^a * a^b (mod p) = 1228773932920826047
   g^M (mod p) = 1228773932920826047

Подпись прошла проверку
```

Рисунок 5 – Запуск программы с параметром $l = 64$

```
E:\5.1\Kript\5\Elgamal\x64\Release>Elgamal.exe 256
Протокол электронной цифровой подписи Эль-Гамаля

Генерация необходимых значений:
   Сгенерировано простое число p = 71483256256004138573351752009162256272212238101729791151341506529799048853463
   Вычислен первообразный корень числа p: g = 2
   Сгенерирован закрытый ключ x = 21570268856108687761791799570039367896456263403821270825755644174110059338503
   Вычислен y = g^x (mod p) = 68872409223459581528603339833171942241215285204503389696166147452611592035413
   Сгенерировано сообщение M = 40289975063617962148031210846573497201726701724753749611646001862556018988980

Подпись сообщения:
   Сгенерировано k взаимнопростое с p-1: k = 43218422185363607346420755878896668284738848789310881946218170105648555644038229
   Вычислен a = g^k (mod p) = 59485149715299276067255242286797643694130867957814033960082096018922993277625
   Вычислен k^(-1) = 61816174271322077628334187959454800213580549205208835420273039632891006558505
   Вычислен b = (M-xa)*k^(-1) (mod p-1) = 31479734706923588741913632884606650655125249636157426756899910994266169857035
   Подписью сообщения M является пара (a, b) = (59485149715299276067255242286797643694130867957814033960082096018922993277625,
 31479734706923588741913632884606650655125249636157426756899910994266169857035)

Проверка подписи:
   y^a * a^b (mod p) = 29891368004984964683492018375969422287468215829569116893775020277856201493797
   g^M (mod p) = 29891368004984964683492018375969422287468215829569116893775020277856201493797

Подпись прошла проверку
```

Рисунок 6 – Запуск программы с параметром $l = 256$

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 << "   Сгенерировано простое число p = " << p << "\n";
      cout << "   Вычислен первообразный корень числа p: g = " << g << "\n";
    cout << "   Сгенерирован закрытый ключ x = " << x << "\n";
    cout << "   Вычислен y = g^x (mod p) = " << y << "\n";
    cout << "   Сгенерировано 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 << "   Сгенерировано 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;
}
```