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

Скрытый канал связи

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 скрытый канал связи на основе DSA.

2 Теоретические сведения

Предположим, Алису и Боба арестовали и посадили в разные тюрьмы. Надзиратель Уолтер позволяет Алисе и Бобу обмениваться сообщениями, но запрещает их шифровать. Уолтер подозревает, что они замышляют побег, поэтому хочет читать всё, что они пишут.

Алисе и Бобу необходимо согласовать свои планы, чтобы сделать это, они должны обмануть надзирателя и найти метод передачи секретной информации. С этой целью они могут под носом Уолтера создать скрытый канал – тайный канал связи, по которому передаются сообщения, сами по себе не содержащие секретную информацию.

Густавус Симмонс выдвинул концепцию создания скрытого канала с помощью обычного алгоритма цифровой подписи: скрытые сообщения спрятаны в том, что выглядит как обычная цифровая подпись. Уолтер видит, как туда и обратно передаются подписанные безобидные сообщения, однако действительно стоящая информация передаётся незаметно для него по скрытому каналу.

По существу, алгоритм подписи, используемый в крытом канале, ничем не отличается от обычного алгоритма создания подписи, по крайней мере, для Уолтера. Последний не только не может читать сообщения, передаваемые по скрытому каналу, но даже не подозревает о его существовании.

В схеме DSA существует несколько скрытых каналов, одни из них позволяет Алисе и Бобу обмениваться в каждой подписи одним битом скрытой информации.

Скрытый канал на основе DSA

Вход: l – битовая длина числа p такая, что $l\ (mod\ 64) = 0$ и $512 \leq l \leq 1024$, s – битовая длина секретного сообщения m'.

Выход: Скрытое сообщение, которое получил Боб.

1) Генерация основных параметров схемы:

Шаг 1. Выбирается простое число q, битовая длина которого равна 160, при этом q удовлетворяет условию: $H(m) < q$, где m – безобидное сообщение, $H(m)$ – 160-ти битовая строка, полученная от m с помощью алгоритма SHA;

Шаг 2. Вычисляется простое число p по формуле: $p = 2^t q + 1$;

Шаг 3. Вычисляется g по формуле: $g = h^{\frac{p-1}{q}} (\mathrm{mod}\ p)$, где h – произвольное число от 2 до $p - 2$ такое, что $g \neq 1$ (в большинстве случаев $h = 2$ удовлетворяет данному условию);

Шаг 4. Вычисляется большое простое число P, отличающееся от p – секретный ключ Алисы и Боба для скрытого канала;

Шаг 5. Выбирается x от 1 до $q - 1$ – закрытый ключ Алисы;

Шаг 6. Вычисляется $y = g^x (mod\ p)$ – открытый ключ Алисы.

Элементы p, q, g, y являются открытыми.

2) Для каждого бита секретного сообщения выполняются пункты 2.1)-2.4).

2.1) Генерация подписи для сообщения m:

Шаг 1. Алиса выбирает k от 1 до $q - 1$;

Шаг 2. Алиса вычисляет r по формуле $r = (g^k (mod\ p))(mod\ q)$, если текущий бит секретного сообщения равен 0, то r должно быть квадратичным невычетом по модулю P, если текущий бит секретного сообщения равен 1, то r должно быть квадратичным вычетом по модулю P. Если данные условия не выполняются или $r = 0$, то перейти на шаг 1 и сгенерировать новое k;

Шаг 3. Алиса вычисляет s по формуле $s = k^{-1} * (H(m) + xr)\ (mod\ q)$. Если $s = 0$, то перейти на шаг 1 и сгенерировать новое k.

2.2) Алиса посылает Бобу сообщение m и подпись (r, s).

2.3) Проверка подписи:

Шаг 1. Боб вычисляет $u = s^{-1} (mod\ q)$;

Шаг 2. Боб вычисляет $a = (H(m) * u)\ (mod\ q)$;

Шаг 3. Боб вычисляет $b = (r * u)\ (mod\ q)$;

Шаг 4. Боб вычисляет $v = (g^a * y^b \bmod p) \pmod q$. Если $v = r$, то подпись прошла проверку, иначе нет.

2.4) Восстановление скрытого бита: если r квадратичный невычет по модулю P, то Боб восстанавливает 0, если r квадратичный вычет по модулю P, то Боб восстанавливает 1.

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 введённых, при запуске программы, битовой длины числа p, битовой длины секретного сообщения m' и количества подробно описанных итераций.

Функция $SecretChannelDSA$ содержит все шаги описанной выше схемы, при этом для генерации простых чисел q, P и получения $H(m)$ используются функции из библиотеки $Crypto++$.

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

Для поиска обратного элемента и НОДа двух чисел в поле используется расширенный алгоритм Евклида – функция $ExtendedEuclid$. Для проверки числа на простоту используется алгоритм Миллера-Рабина, который реализован в функции $MilRab$. Генерация безобидного и секретного сообщений происходит в функции $GenMess$. Проверка на квадратичный вычет и невычет выполняется с помощью символа Якоби – функция Jac. Кроме того, в программе реализована функция быстрого возведения в степень по модулю, которая используется в большом количестве шагов – функция $Exponentiation$.

3.2 Результаты тестирования программы

Отсутствие одного или нескольких параметров, при запуске программы, ведёт к соответствующей ошибке, данный запуск представлен на рисунке 1.

```
E:\5.1\Kript\7\SecretChannelDSA\x64\Release>SecretChannelDSA.exe

   Ошибка: Недостаточное количество параметров
   На входе должны быть следующие параметры:
      l - битовая длина числа p
      s - битовая длина секретного сообщения
      i - сколько итераций необходимо подробно описать
```

Рисунок 1 – Запуск программы без параметров

Число l должно делиться на 64 и быть в диапазоне от 512 до 1024 – рисунок 2.

```
E:\5.1\Kript\7\SecretChannelDSA\x64\Release>SecretChannelDSA.exe l=66 s=10 i=10

   Ошибка в параметре l: должно быть в диапозоне от 512 до 1024 и делиться на 64
```

Рисунок 2 – Запуск программы с параметром l, не удовлетворяющим условиям

Число s должно быть больше 0 – рисунок 3.

```
E:\5.1\Kript\7\SecretChannelDSA\x64\Release>SecretChannelDSA.exe l=512 s=0 i=10

   Ошибка в параметре s: должно быть больше 0
```

Рисунок 3 – Запуск программы с параметром s, меньшим 1

Число i должно быть в промежутке от 0 до s – рисунок 4.

```
E:\5.1\Kript\7\SecretChannelDSA\x64\Release>SecretChannelDSA.exe l=512 s=10 i=11

   Ошибка в параметре i: должно быть в диапозоне от 0 до 10
```

Рисунок 4 – Запуск программы с параметром i, больше s

На рисунках 5-6 представлен успешный запуск программы, в котором подробно выводятся две итерации отправки бит секретного сообщения из десяти.

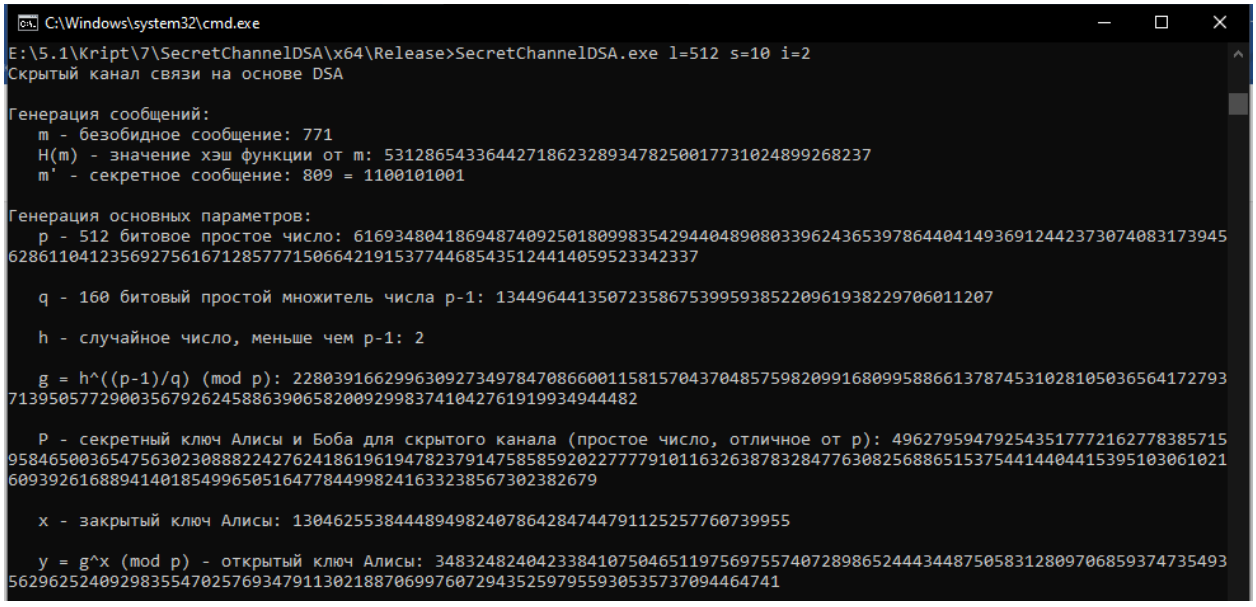

Рисунок 5 – Запуск программы с подробным выводом 2-х итераций из 10-ти (часть1)

```
Передача секретного сообщения:

   Генерация подписи Алисой:
      Генерация случайного числа k = 4776675965113734381983598355100440906541869562 46
      Вычисление r = (g^k mod p) (mod q) = 760075389611842905027861720089142850361004065941
      Вычисление s = (k^-1 * (H(m) + xr)) (mod q) = 4574415680650725959793019951451550467641700919 45

   Алиса передаёт вместе с Уолтером сообщение m и подпись (r,s)

   Проверка подписи Уолтером:
      Вычисление u = s^-1 (mod q) = 1043680597056870897185266555580830916018023241986
      Вычисление a = H(m) * u (mod q) = 3736714256170163661707235386303548383392026995 48
      Вычисление b = r * u (mod q) = 6180233883118073820904106730692904556942911787 60
      Вычисление v = (g^a * y^b mod p) mod q = 760075389611842905027861720089142850361004065941
      Уолтер убедился, в подлинности сообщения (r = v)

   Уолтер передал Бобу сообщение и подпись Алисы

   Проверка подписи Бобом:
      Вычисление u = s^-1 (mod q) = 1043680597056870897185266555580830916018023241986
      Вычисление a = H(m) * u (mod q) = 3736714256170163661707235386303548383392026995 48
      Вычисление b = r * u (mod q) = 6180233883118073820904106730692904556942911787 60
      Вычисление v = (g^a * y^b mod p) mod q = 760075389611842905027861720089142850361004065941
      Боб убедился, в подлинности сообщения (r = v)

   Алиса отправила 1, Боб получил 1
```

Рисунок 6 – Запуск программы с подробным выводом 2-х итераций из 10-ти (часть 2)

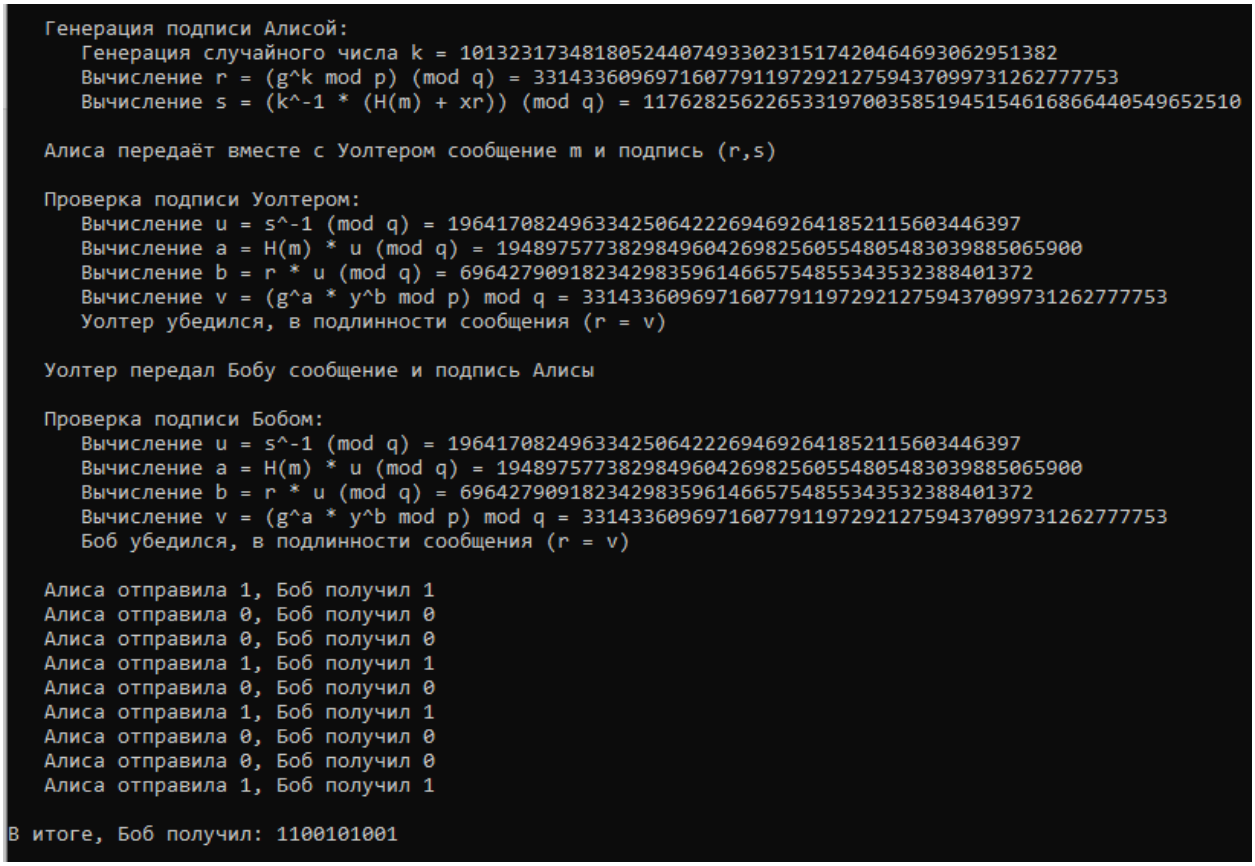

Рисунок 7 – Запуск программы с подробным выводом 2-х итераций из 10-ти (часть 3)

Листинг кода

```
#include <iostream>
#include <string>
#include "osrng.h"
#include "dsa.h"
#include "dh.h"
#include <random>
#include "files.h"
#include <set>
#include <hex.h>
#include <boost/multiprecision/cpp_int.hpp>
#include <boost/random/uniform_int.hpp>
#include <boost/random/variate_generator.hpp>
#include <boost/multiprecision/cpp_dec_float.hpp>

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

AutoSeededRandomPool rng;

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, int k) {
    cpp_int N = n, Y = 1, Z = x;
    if (k == 1)
        Z = Z % m;
    else
        N--;
    while (N != 0) {
        cpp_int lastN = N % 2;
        N = N / 2;
        if (lastN == 0) {
            Z = (Z * Z);
            if (k == 1)
                Z = Z % m;
            continue;
        }
        Y = (Y * Z);
```

```
            if (k == 1)
                Y = Y % m;
            if (N == 0)
                break;
            Z = (Z * Z);
            if (k == 1)
                Z = Z % m;
        }
        if (k == 1)
            Y = Y % m;
        return Y;
    }

    cpp_int Jac(cpp_int a, cpp_int b) {
        if (ExtendedEuclid(a, b)[0] != 1)
            return 0;
        else {
            int r = 1;
            while (a != 0) {
                cpp_int t = 0;
                while (a % 2 == 0) {
                    t = t + 1;
                    a = a / 2;
                }
                if (t % 2 != 0)
                    if (Exponentiation(b, 1, 8, 1) == 3 || Exponentiation(b,
1, 8, 1) == 5)
                        r = r * (-1);
                if (Exponentiation(a, 1, 4, 1) == 3 && Exponentiation(b, 1,
4, 1) == 3)
                    r = r * (-1);
                cpp_int c = a;
                if (c != 0)
                    a = Exponentiation(b, 1, c, 1);
                b = c;
            }
            return r;
        }
    }

    bool MilRab(cpp_int p, cpp_int k) {
        if (p == 1 || p == 2 || p == 3)
            return true;
        if (p % 2 == 0)
            return false;
        cpp_int t = p - 1;
        cpp_int s = 0;
        while (t % 2 == 0) {
            t = t / 2;
            s++;
        }
        for (cpp_int i = 0; i < k; i++) {
            cpp_int a = rand() % (p - 3) + 2;
            if (ExtendedEuclid(p, a)[0] > 1)
                return false;
            cpp_int x = Exponentiation(a, t, p, 1);
```

```
        if (x == 1 || x == p - 1)
            continue;
        for (cpp_int g = 1; g < s; g++) {
            x = x * x % p;
            if (x == 1)
                return false;
            if (x == p - 1)
                break;
        }
        if (x != p - 1)
            return false;
    }
    return true;
}

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;
}

cpp_int GenMess(int l) {
    cpp_int m = 1;
    cpp_int deg = 2;
    for (int i = 1; i < l - 1; i++) {
        m = m + (deg * (rand() % 2));
        deg = deg * 2;
    }
    m = m + deg;
    return m;
}

cpp_int toSHA1(string mess) {
    string digest, res;
    SHA1 hash;
    hash.Update((const byte*)mess.data(), mess.size());
    digest.resize(hash.DigestSize());
    hash.Final((byte*)&digest[0]);
    StringSource(digest, true, new HexEncoder(new StringSink(res)));
    istringstream stream(res);
    cpp_int dec;
    stream >> hex >> dec;
    return dec;
}

vector <cpp_int> toBits(cpp_int n) {
    vector <cpp_int> res1, res2;
```

```
        while (n > 1) {
            res1.push_back(n % 2);
            n = n / 2;
        }
        res1.push_back(n);
        for (int i = res1.size() - 1; i > -1; i--)
            res2.push_back(res1[i]);
        return res2;
    }

    bool checkSignature(cpp_int s, cpp_int q, cpp_int H, cpp_int r, cpp_int
g, cpp_int p, cpp_int y, bool check, string str) {
        cpp_int u = ModNegative(ExtendedEuclid(s, q)[1], q);
        cpp_int a = (H * u) % q;
        cpp_int b = (r * u) % q;
        cpp_int prom1 = Exponentiation(g, a, p, 1);
        cpp_int prom2 = Exponentiation(y, b, p, 1);
        cpp_int v = ((prom1 * prom2) % p) % q;
        if (check) {
            cout << "\n   Проверка подписи " << str << "ом:\n";
            cout << "      Вычисление u = s^-1 (mod q) = " << u << "\n";
            cout << "      Вычисление a = H(m) * u (mod q) = " << a << "\n";
            cout << "      Вычисление b = r * u (mod q) = " << b << "\n";
            cout << "       Вычисление v = (g^a * y^b mod p) mod q = " << v
<< "\n";
            if (r == v)
                cout << "     " << str << " убедился, в подлинности сообщения
(r = v)\n";
        }
        if (r == v)
            return true;
        return false;
    }

    void SecretChannelDSA(int l, int sizem, int iter) {

        //Сообщения
        cpp_int m, msh, H;
        m = GenMess(sizem);
        H = toSHA1(to_string(m));
        msh = GenMess(sizem);
        vector <cpp_int> bitsMsh = toBits(msh);
        cout << "\nГенерация сообщений:\n";
        cout << "   m - безобидное сообщение: " << m;
        cout << "\n   H(m) - значение хэш функции от m: " << H;
        cout << "\n   m' - секретное сообщение: " << msh << " = ";
        for (int i = 0; i < bitsMsh.size(); i++)
            cout << bitsMsh[i];

        //p и q
        DH dh1; cpp_int p , q;
        cpp_int deg = Exponentiation(2, l - 160, 1, 2);
        for (;;) {
            do {
                dh1.AccessGroupParameters().GenerateRandomWithKeySize(rng,
160);
```

```
                q = IntegerToCppint(dh1.GetGroupParameters().GetModulus());
            } while (H >= q);
            p = deg * q + 1;
            if (MilRab(p, 10))
                break;
        }
        cout << "\n\nГенерация основных параметров:\n";
        cout << "   p - " << l << " битовое простое число: " << p;
        cout << "\n\n   q - " << 160 << " битовый простой множитель числа p-
1: " << q;

        //генерация h и g
        cpp_int h = 2, g;
        do {
            g = Exponentiation(h, (p - 1) / q, p, 1);
            if (Exponentiation(g, q, p, 1) != 1) {
                g = 0;
                h = Random(2, p - 2);
                continue;
            }
        } while (g < 2);
        cout << "\n\n   h - случайное число, меньше чем p-1: " << h;
        cout << "\n\n   g = h^((p-1)/q) (mod p): " << g << "\n";

        //генерация P
        cpp_int P = p;
        DH dh;
        while (p == P) {
            dh.AccessGroupParameters().GenerateRandomWithKeySize(rng,
rand() % 1020 + 4);
            P = IntegerToCppint(dh.GetGroupParameters().GetModulus());
        }
        cout << "\n   P - секретный ключ Алисы и Боба для скрытого канала
(простое число, отличное от p): " << P;

        //открытый и закрытый ключ Алисы
        DH dh2;
        cpp_int x, y;
        x = Random(1, q - 1);
        y = Exponentiation(g, x, p, 1);
        cout << "\n\n   x - закрытый ключ Алисы: " << x;
        cout << "\n\n    y = g^x (mod p) - открытый ключ Алисы: " << y <<
"\n";

        //Протокол
        cout << "\nПередача секретного сообщения:";
        iter--;
        vector <cpp_int> bitsBob;
        for (int i = 0; i < bitsMsh.size();) {
            //Алиса
            cpp_int k = Random(1, q - 1);
            if (ExtendedEuclid(k, q)[0] != 1)
                continue;
            cpp_int r = Exponentiation(g, k, p, 1) % q;
            if (bitsMsh[i] == 1) {
                if (Jac(r, P) != 1)
```

```
                    continue;
                }
                else{
                    if (Jac(r, P) != -1)
                        continue;
                }
                if (r == 0)
                    continue;
                cpp_int s = (ModNegative(ExtendedEuclid(k, q)[1], q) * (H + x *
r)) % q;
                if (s == 0 || ExtendedEuclid(s, q)[0] != 1)
                    continue;
                bool checkIter = iter >= i;
                if (checkIter) {
                    cout << "\n\n   Генерация подписи Алисой:";
                    cout << "\n      Генерация случайного числа k = " << k;
                    cout << "\n       Вычисление r = (g^k mod p) (mod q) = " <<
r;
                    cout << "\n      Вычисление s = (k^-1 * (H(m) + xr)) (mod q)
= " << s;
                    cout << "\n\n   Алиса передаёт вместе с Уолтером сообщение m
и подпись (r,s)\n";
                }
                //Уолтер
                bool checkSignW = checkSignature(s, q, H, r, g, p, y, iter >= i,
"Уолтер");
                if (checkSignW) {
                    if (checkIter)
                        cout << "\n    Уолтер передал Бобу сообщение и подпись
Алисы\n";
                    //Боб
                    bool checkSignB = checkSignature(s, q, H, r, g, p, y, iter
>= i, "Боб");
                    if (checkSignB) {
                        cpp_int quad = Jac(r, P);
                        cpp_int bitBob;
                        if (quad == 1)
                            bitBob = 1;
                        if (quad == -1)
                            bitBob = 0;
                        cout << "\n   Алиса отправила " << bitsMsh[i] << ", Боб
получил " << bitBob;
                        bitsBob.push_back(bitBob);
                    }
                }
                i++;
            }
            cout << "\n\nВ итоге, Боб получил: ";
            for (int i = 0; i < bitsBob.size(); i++)
                cout << bitsBob[i];
            cout << "\n";
        }


        void ErrMess() {
            cerr << "   На входе должны быть следующие параметры:\n"
```

```
            << "      l - битовая длина числа p\n"
            << "      s - битовая длина секретного сообщения\n"
            << "      i - сколько итераций необходимо подробно описать\n";
    }

    int main(int argc, char** argv) {
        setlocale(LC_ALL, "Russian");
        srand(time(NULL));
        if (argc < 2) {
            cerr << "\n   Ошибка: Недостаточное количество параметров\n";
            ErrMess();
            return 0;
        }
        int s = -1, l = -1, iter = -1;
        string prom1, prom2;
        for (int i = 1; i < argc; i++) {
            prom1 = argv[i];
            try {
                prom1.erase(2, prom1.size() - 2);
                if (prom1 == "l=") {
                    prom2 = argv[i];
                    prom2.erase(0, 2);
                    try {
                        l = stoi(prom2);
                    }
                    catch (exception) {
                        cerr << "\n     Ошибка в параметре l: передано
некорректное число\n";
                        ErrMess();
                        return 0;
                    }
                    continue;
                }
                if (prom1 == "s=") {
                    prom2 = argv[i];
                    prom2.erase(0, 2);
                    try {
                        s = stoi(prom2);
                    }
                    catch (exception) {
                        cerr << "\n     Ошибка в параметре s: передано
некорректное число\n";
                        ErrMess();
                        return 0;
                    }
                    continue;
                }
                if (prom1 == "i=") {
                    prom2 = argv[i];
                    prom2.erase(0, 2);
                    try {
                        iter = stoi(prom2);
                    }
                    catch (exception) {
                        cerr << "\n     Ошибка в параметре i: передано
некорректное число\n";
```

```
                        ErrMess();
                        return 0;
                    }
                    continue;
                }
            }
            catch (exception) {
                continue;
            }
        }
        if (l < 0 || s < 0 || iter < 0) {
            cerr << "\n     Ошибка: некоторые параметры отсутствуют или
пустые\n";
            ErrMess();
            return 0;
        }
        if (l < 512 || l > 1024 || l % 64 != 0) {
            cerr << "\n   Ошибка в параметре l: должно быть в диапозоне от
512 до 1024 и делиться на 64\n";
            return 0;
        }
        if (s < 1) {
            cerr << "\n   Ошибка в параметре s: должно быть больше 0\n";
            return 0;
        }
        if (iter < 0 || iter > s) {
            cerr << "\n   Ошибка в параметре i: должно быть в диапозоне от 0
до " << s << "\n";
            return 0;
        }
        cout << "Скрытый канал связи на основе DSA\n";
        SecretChannelDSA(l, s, iter);
        return 0;
    }
```