

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

Разделение секрета

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

подпись, дата

Р. А. Фарахутдинов

Саратов 2023

1 Постановка задачи

Необходимо реализовать схему разделения секрета Асмута-Блума, на основе греко-римской теореме об остатках.

2 Теоретические сведения

Разделение секрета (англ. *secret sharing*) — термин в криптографии, под которым понимают любой из способов распределения секрета среди группы участников, каждому из которых достаётся своя некая доля. Секрет может воссоздать только коалиция участников из первоначальной группы, причём входить в коалицию должно не менее некоторого изначально известного их числа.

Схемы разделения секрета применяются в случаях, когда существует значимая вероятность компрометации одного или нескольких хранителей секрета, но вероятность недобросовестного сговора значительной части участников считается пренебрежимо малой.

Существующие схемы имеют две составляющие: разделение и восстановление секрета. К разделению относится формирование частей секрета и распределение их между членами группы, что позволяет разделить ответственность за секрет между её участниками. Обратная схема должна обеспечить его восстановление при условии доступности его хранителей в некотором необходимом количестве.

Схема Асмута-Блума – пороговая схема разделения секрета, построенная с использованием простых чисел. Позволяет разделить секрет M между n сторонами таким образом, что его смогут восстановить любые t участников.

Схема Асмута-Блума

Вход: Секрет M , число участников n и минимальное число участников t , нужное для восстановления секрета.

Выход: Результат попытки восстановления секрета.

Этап 1 – Разделение секрета

Шаг 1. Выбирается простое число p , большее M ;

Шаг 2. Выбирается n взаимно простых друг с другом чисел $d_1 \dots d_n$, таких что:

- $\forall i$ выполняется $d_i > p$;
- $\forall i$ выполняется $d_i < d_{i+1}$;
- $d_1 * d_2 * \dots * d_m > p * d_{n-m+2} * d_{n-m+3} * \dots * d_n$;

Шаг 3. Выбирается случайное число r и вычисляется $M' = M + r * p$, такое что $M' > d_1 * d_2 * \dots * d_{m-1}$;

Шаг 4. Вычисляются доли $k_i = M' \pmod{d_i}$;

Шаг 5. Участникам раздаются $\{p, d_i, k_i\}$.

Этап 2 – Восстановление секрета

Шаг 1. Выбирается m или более участников, обозначим за $\{m_1, m_2, \dots, m_m, \dots\}$;

Шаг 2. Вычисляются обратные элементы $k_{m_i}^{-1} = (k_{m_i})^{-1} \pmod{d_{m_i}}$;

Шаг 3. Вычисляется $Mult = \prod d_{m_i}$;

Шаг 3. С помощью теоремы вычисляется $(M')^{-1}$ из следующей системы сравнений по модулю $Mult$:

$$\begin{cases} (M')^{-1} \equiv k_{m_1}^{-1} \pmod{d_{m_1}} \\ (M')^{-1} \equiv k_{m_2}^{-1} \pmod{d_{m_2}} \\ \dots \\ (M')^{-1} \equiv k_{m_m}^{-1} \pmod{d_{m_m}} \\ \dots \end{cases}$$

Шаг 4. Вычисляется $M = ((M')^{-1})^{-1} \pmod{Mult} \pmod{p}$.

Если участвовали участники из первого этапа и их было больше, чем $m - 1$, то они получают тот же секрет M , что был изначально, иначе нет.

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 введенных, при запуске программы, секрета M , числа участников n и минимального числа участников m , нужного для восстановления секрета.

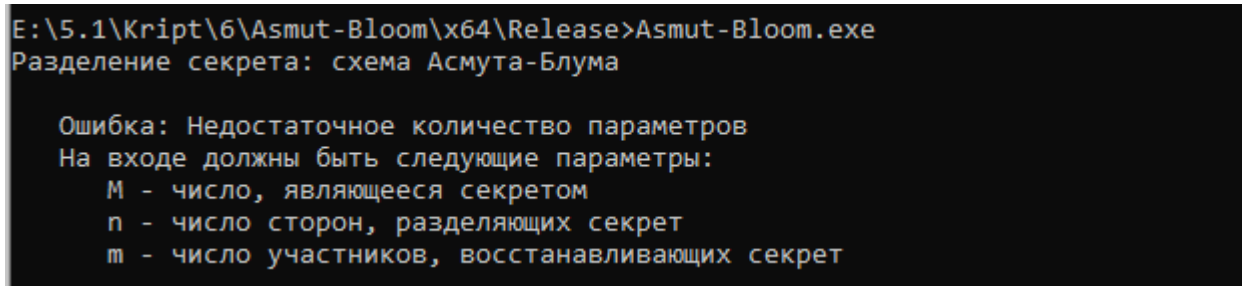
Функция *AsmutBloom* содержит все шаги описанной выше схемы, при этом для генерации простых чисел p и d_i используется функция из библиотеки *Crypto++*.

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

Для поиска обратного элемента в поле используется расширенный алгоритм Евклида – функция *ExtendedEuclid*. А нахождение решения системы сравнения по модулю реализовано в функции *SystemTh*.

3.2 Результаты тестирования программы

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

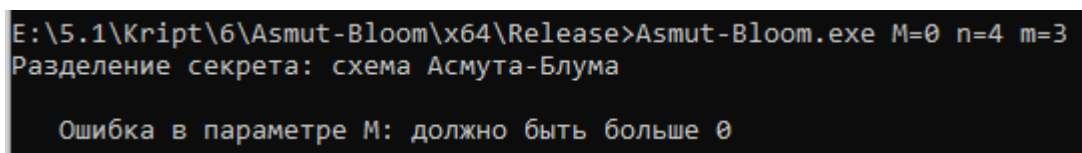


```
E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe
Разделение секрета: схема Асмута-Блума

Ошибка: Недостаточное количество параметров
На входе должны быть следующие параметры:
  М - число, являющееся секретом
  n - число сторон, разделяющих секрет
  m - число участников, восстанавливающих секрет
```

Рисунок 1 – Запуск программы без параметров

M должно быть больше 0 – рисунок 2.



```
E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe M=0 n=4 m=3
Разделение секрета: схема Асмута-Блума

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

Рисунок 2 – Запуск программы с параметром M меньшим 1

Число участников n должно быть больше 1 – рисунок 3.

```
E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe M=4 n=1 m=3
Разделение секрета: схема Асмута-Блума

Ошибка в параметре n: должно быть больше 1
```

Рисунок 3 – Запуск программы с параметром n меньшим 2

Число m должно быть в промежутке от 2 до n – рисунок 4.

```
E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe M=4 n=5 m=1
Разделение секрета: схема Асмута-Блума

Ошибка в параметре m: должно быть больше 1 и не превосходить n

E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe M=4 n=5 m=7
Разделение секрета: схема Асмута-Блума

Ошибка в параметре m: должно быть больше 1 и не превосходить n
```

Рисунок 4 – Запуск программы с параметром m меньшим 2 или больше n

На рисунках 5-6 представлены успешные запуски программ, доказывающие, что m участников могут восстановить секрет, а $m - 1$ – нет.

```
E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe M=2 n=4 m=3
Разделение секрета: схема Асмута-Блума

Секрет M = 2
Число участников n = 4
Минимальное число участников, нужное для восстановления секрета, m = 3

Было сгенерированно p = 33889
Было сгенерированно g = 885892428
Было вычисленно M' = M + gp = 30022008492494
После выбора di и вычисления долей ki = M' mod di участникам были разданы {p, di, ki}:
Участник 1: {33889, 54881, 44123}
Участник 2: {33889, 55051, 32882}
Участник 3: {33889, 60899, 45713}
Участник 4: {33889, 62423, 43737}

Запустить дефолтное (m = 3, участники выбираются случайно) восстановление секрета? [y,n]: y
Участники 1 3 4 хотят восстановить секрет
Участники получили M' = 30022008492494
Участники вычислили M = M' (mod p) = 2

M, разделённое в начале, сопало с M, полученным в конце

Запустить дефолтное (m = 3, участники выбираются случайно) восстановление секрета? [y,n]: n
Введите новое m: 2
Введите разных участников от 1 до 4: 1 4
Участники 1 4 хотят восстановить секрет
Участники получили M' = 1401814625
Участники вычислили M = M' (mod p) = 30029

M, разделённое в начале, не сопало с M, полученным в конце
```

Рисунок 5 – Запуск программы с корректными параметрами

```

E:\5.1\Kript\6\Asmut-Bloom\x64\Release>Asmut-Bloom.exe M=2314512 n=10 m=5
Разделение секрета: схема Асмута-Блума

Секрет M = 2314512
Число участников n = 10
Минимальное число участников, нужное для восстановления секрета, m = 5

Было сгенерированно p = 4944523
Было сгенерированно g = 1573469552165128956036328851
Было вычисленно M' = M + gp = 7780056390480179921087616841647585
После выбора di и вычисления долей ki = M' mod di участникам были разданы {p, di, ki}:
Участник 1: {4944523, 6132271, 2820358}
Участник 2: {4944523, 6147277, 5270833}
Участник 3: {4944523, 6399947, 689373}
Участник 4: {4944523, 6658079, 5509974}
Участник 5: {4944523, 6767437, 2600893}
Участник 6: {4944523, 7540801, 1402596}
Участник 7: {4944523, 7544347, 6699567}
Участник 8: {4944523, 7553921, 5376133}
Участник 9: {4944523, 7737767, 5718855}
Участник 10: {4944523, 7947361, 4773101}

Запустить дефолтное (m = 5, участники выбираются случайно) восстановление секрета? [y,n]: y
Участники 1 5 6 9 10 хотят восстановить секрет
Участники получили M' = 7780056390480179921087616841647585
Участники вычислили M = M'(mod p) = 2314512

M, разделённое в начале, сопало с M, полученным в конце

Запустить дефолтное (m = 5, участники выбираются случайно) восстановление секрета? [y,n]: n
Введите новое m: 4
Введите разных участников от 1 до 10: 1 5 8 10
Участники 1 5 8 10 хотят восстановить секрет
Участники получили M' = 2461059722292455050616314951
Участники вычислили M = M'(mod p) = 2209027

M, разделённое в начале, не сопало с M, полученным в конце

```

Рисунок 6 – Запуск программы с корректными параметрами

Листинг кода

```
#include <iostream>
#include <string>
#include "osrng.h"
#include "rsa.h"
#include <random>
#include <set>
#include <boost/multiprecision/cpp_int.hpp>
#include <boost/random/uniform_int.hpp>
#include <boost/random/variante_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 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;
    }

int BitLength(cpp_int M) {
    cpp_dec_float_50 prom = log2(cpp_dec_float_50(M) + 1);
    cpp_int res(prom);
    if (prom - cpp_dec_float_50(res) != 0)
        res++;
    string s = to_string(res);
    return stoi(s);
}

cpp_int SystemTh(vector <cpp_int> ms, vector <cpp_int> us, cpp_int M) {
    cpp_int u = 0;
    vector <cpp_int> c, d;
    for (int i = 0; i < ms.size(); i++) {
        c.push_back(M / ms[i]);
        d.push_back(ExtendedEuclid(c[i], ms[i])[1]);
        u = (u + c[i] * d[i] * us[i]);
        u = ModNegative(u, M) % M;
    }
    return u;
}

void AsmutBloom(cpp_int M, int n, int m) {
    cout << "\n    Секрет M = " << M << "\n";
    cout << "    Число участников n = " << n << "\n";
    cout << "    Минимальное число участников, нужное для восстановления секрета, m = " << m << "\n";

    //Генерация p
    int sizeM = BitLength(M);
    if (sizeM < 16)
        sizeM = 15;
    cpp_int p;
    InvertibleRSAFunction params;
    params.GenerateRandomWithKeySize(rng, (sizeM * 2) + 1);
    p = IntegerToCppint(params.GetPrime1());
    cout << "\n    Было сгенерированно p = " << p << "\n";

    //Генерация di
    bool checker = true;
    set<cpp_int> diSet;
    vector<cpp_int> diVector;
    while (checker) {
        diSet.clear();
        diVector.clear();

        // di > p и di < di+1
        while (diSet.size() < n) {
            params.GenerateRandomWithKeySize(rng, (sizeM * 2) + 2);
            diSet.insert(IntegerToCppint(params.GetPrime1()));
        }
    }
}

```



```

//Проверка последнего условия
cpp_int m1 = m;
cpp_int left = 1, right = p;
for (cpp_int di : diSet)
    diVector.push_back(di);
for (int i = 0; i < diVector.size(); i++) {
    if (m1 == 0) {
        right = right * diVector[i];
        continue;
    }
    left = left * diVector[i];
    m1--;
    if (m1 == 0)
        i = diVector.size() - m + 1;
}
if (left > right)
    checker = false;
}

//Генерация r и вычисление M'
cpp_int diMult = 1;
for (int i = 0; i < m - 1; i++)
    diMult = diMult * diVector[i];
cpp_int r, Msh;
do {
    r = Random(2, diMult);
    Msh = M + r * p;
} while (Msh <= diMult);
cout << "    Было сгенерированно r = " << r << "\n";
cout << "    Было вычисленно M' = M + rp = " << Msh << "\n";

//Вычисление долей ki
vector<cpp_int> kiVector;
for (int i = 0; i < diVector.size(); i++)
    kiVector.push_back(Msh % diVector[i]);

cout << "    После выбора di и вычисления долей ki = M' mod di участникам
были разданы {p, di, ki}:\n";
for (int i = 0; i < diVector.size(); i++)
    cout << "        Участник " << i + 1 << ": {" << p << ", " <<
diVector[i] << ", " << kiVector[i] << "}\n";

//Восстановление секрета
string answer;
for (;;) {
    cout << "\n    Запустить дефолтное (m = " << m << ", участники
выбираются случайно) восстановление секрета? [y,n]: ";
    cin >> answer;
    if (answer != "n" && answer != "y")
        break;
    set<int> participants;
    vector<cpp_int> diUsed, kiUsed;
    cpp_int diMultUsed = 1, MshSekret, MSekret;
    int newm, participant;
    if (answer == "y") {

```

```

        while (participants.size() != m)
            participants.insert(rand() % n);
    }
    else {
        for (;;) {
            cout << "        Введите новое m: ";
            try {
                cin >> newm;
            }
            catch (exception) {
                cerr << "\n    Введены некорректные данные";
                continue;
            }
            if (newm > 1 && newm < n + 1)
                break;
        }
        cout << "        Введите разных участников от 1 до " << n << ": ";
        while (participants.size() != newm) {
            try {
                cin >> participant;
            }
            catch (exception) {
                cerr << "\n    Введены некорректные данные";
                continue;
            }
            if (participant < 0 || newm > n)
                continue;
            participants.insert(participant - 1);
        }
    }
    cout << "        Участники ";
    for (int i : participants) {
        cout << i + 1 << " ";
        diUsed.push_back(diVector[i]);
        kiUsed.push_back(ModNegative(ExtendedEuclid(kiVector[i],
diVector[i])[1], diVector[i]));
    }
    cout << "хотят восстановить секрет\n";
    for (int i = 0; i < diUsed.size(); i++)
        diMultUsed = diMultUsed * diUsed[i];
    MshSekret = SystemTh(diUsed, kiUsed, diMultUsed);
    MshSekret = ModNegative(ExtendedEuclid(MshSekret, diMultUsed)[1],
diMultUsed);
    cout << "        Участники получили M' = " << MshSekret << "\n";
    MSekret = MshSekret % p;
    cout << "        Участники вычислили M = M'(mod p) = " << MSekret <<
"\n";

    if (M == MSekret)
        cout << "\n    M, разделённое в начале, сопало с M, полученным в
конце\n";
    else
        cout << "\n    M, разделённое в начале, не сопало с M, полученным
в конце\n";
    }
}

```

```

void ErrMess() {
    cerr << "    На входе должны быть следующие параметры:\n"
        << "        М - число, являющееся секретом\n"
        << "        n - число сторон, разделяющих секрет\n"
        << "        m - число участников, восстанавливающих секрет\n";
}

int main(int argc, char** argv) {
    setlocale(LC_ALL, "Russian");
    int l;
    cout << "Разделение секрета: схема Асмута-Блума\n";
    srand(time(NULL));
    if (argc < 4) {
        cerr << "\n    Ошибка: Недостаточное количество параметров\n";
        ErrMess();
        return 0;
    }
    cpp_int M = -1;
    int m = -1, n = -1;
    string prom1, prom2;
    for (int i = 1; i < argc; i++) {
        prom1 = argv[i];
        try {
            prom1.erase(2, prom1.size() - 2);
            if (prom1 == "M=") {
                prom2 = argv[i];
                prom2.erase(0, 2);
                try {
                    cpp_int promM(prom2);
                    M = promM;
                }
                catch (exception) {
                    cerr << "\n    Ошибка в параметре М: передано некорректное
число\n";

                    ErrMess();
                    return 0;
                }
                continue;
            }
            if (prom1 == "n=") {
                prom2 = argv[i];
                prom2.erase(0, 2);
                try {
                    n = stoi(prom2);
                }
                catch (exception) {
                    cerr << "\n    Ошибка в параметре n: передано некорректное
число\n";

                    ErrMess();
                    return 0;
                }
                continue;
            }
            if (prom1 == "m=") {

```

```

        prom2 = argv[i];
        prom2.erase(0, 2);
        try {
            m = stoi(prom2);
        }
        catch (exception) {
            cerr << "\n  Ошибка в параметре m: передано некорректное
число\n";
            ErrMess();
            return 0;
        }
        continue;
    }
}
catch (exception) {
    continue;
}
}
if (M < 0 || m < 0 || n < 0) {
    cerr << "\n  Ошибка: некоторые параметры отсутствуют или пустые\n";
    ErrMess();
    return 0;
}
if (M < 1) {
    cerr << "\n  Ошибка в параметре M: должно быть больше 0\n";
    return 0;
}
if (n < 2) {
    cerr << "\n  Ошибка в параметре n: должно быть больше 1\n";
    return 0;
}
if (m < 2 || m > n) {
    cerr << "\n  Ошибка в параметре m: должно быть больше 1 и
непревосходить n\n";
    return 0;
}
AsmutBloom(M, n, m);
return 0;
}

```