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 | \_\_\_\_\_\_\_\_\_\_ | Р. А. Фарахутдинов |
|  | подпись, дата |  |

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 между сторонами таким образом, что его смогут восстановить любые участников.

**Схема Асмута-Блума**

*Вход*: Секрет , число участников и минимальное число участников , нужное для восстановления секрета.

*Выход*: Результат попытки восстановления секрета.

**Этап 1 – Разделение секрета**

Шаг 1. Выбирается простое число , большее ;

Шаг 2. Выбирается взаимно простых друг с другом чисел , таких что:

* выполняется ;
* выполняется ;
* ;

Шаг 3. Выбирается случайное число и вычисляется , такое что ;

Шаг 4. Вычисляются доли ;

Шаг 5. Участникам раздаются .

**Этап 2 – Восстановление секрета**

Шаг 1. Выбирается или более участников, обозначим за ;n = p ⋅ q жЖ

Шаг 2. Вычисляются обратные элементы ;

Шаг 3. Вычисляется ;

Шаг 3. С помощью теоремы вычисляется из следующей системы сравнений по модулю :

Шаг 4. Вычисляется .

Если участвовали участники из первого этапа и их было больше, чем , то они получат тот же секрет , что был изначально, иначе нет.

**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 введённых, при запуске программы, секрета , числа участников и минимального числа участников , нужного для восстановления секрета.

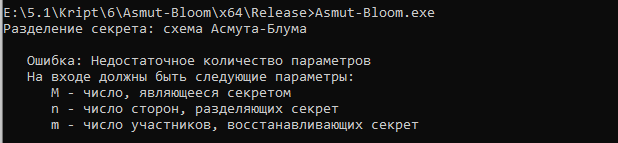
Функция содержит все шаги описанной выше схемы, при этом для генерации простых чисел и используется функция из библиотеки .

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

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

**3.2 Результаты тестирования программы**

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

Рисунок 1 – Запуск программы без параметров

должно быть больше 0 – рисунок 2.

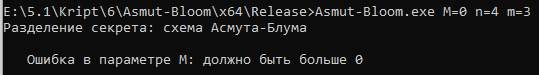


Рисунок 2 – Запуск программы с параметром меньшим 1

Число участников должно быть больше 1 – рисунок 3.

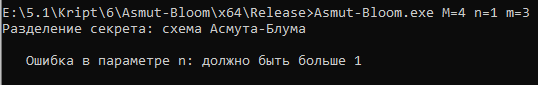


Рисунок 3 – Запуск программы с параметром меньшим 2

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

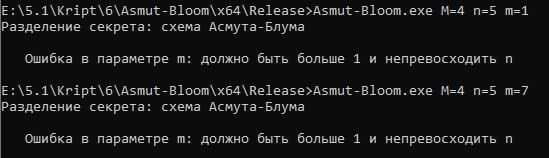


Рисунок – Запуск программы с параметром меньшим 2 или больше

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

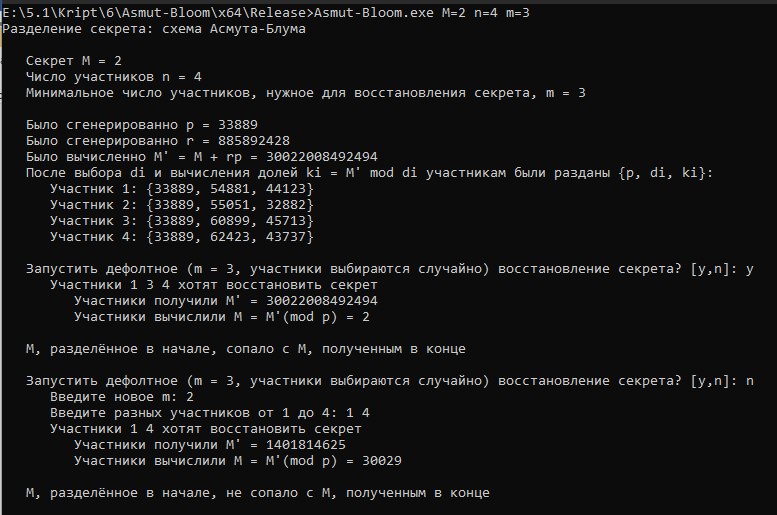


Рисунок 5 – Запуск программы с корректными параметрами

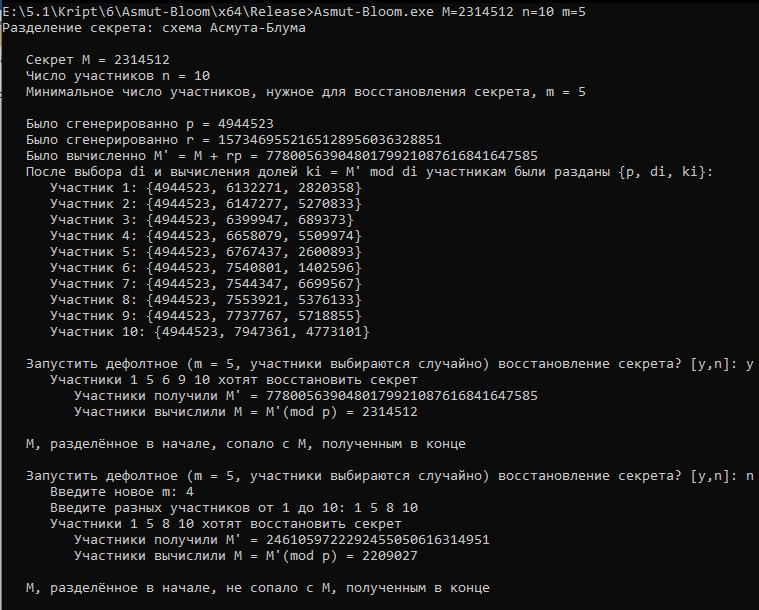


Рисунок 6 – Запуск программы с корректными параметрами

**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/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 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"

<< " M - 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 M: 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;

}