Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра ПМиК

Лабораторная работа №2

«Реализация протоколов Диффи-Хеллман и MQV на эллиптической кривой»

Выполнил:
студент гр. МГ-211 _____ / Бурдуковский И.А./
подпись

Проверил:
Профессор
кафедры ПМиК _____ / Фионов А.Н./

Новосибирск

2023 г.

ОГЛАВЛЕНИЕ

Задание

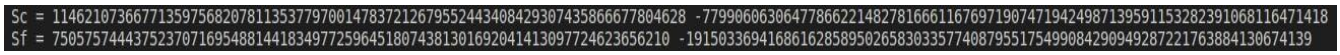
1. Реализовать алгоритмы Диффи–Хеллмана и MQV на эллиптической кривой в аффинном представлении точек.
2. Реализовать алгоритмы Диффи–Хеллмана и MQV на эллиптической кривой в проективном представлении точек.
3. Осуществить замеры времени (в виде числа процессорных циклов) при выполнении основных этапов во всех алгоритмах и провести их сопоставление

Выполнение

1. В аффинном представлении:

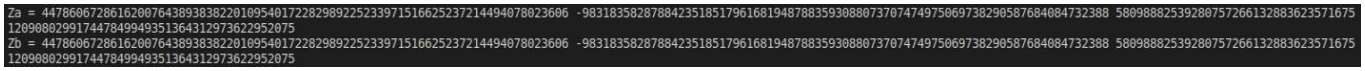
Диффи-Хеллман:

MQV:

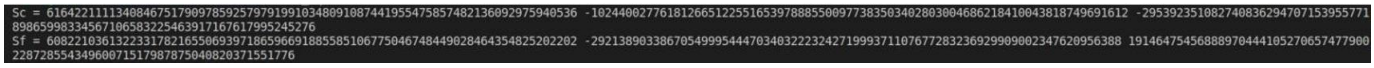


2. В проективном представлении:

Диффи-Хеллман:



MQV:



3. Замеры числа процессорных циклов.

	Число процессорных циклов
оритм Диффи-Хелмана на эллиптической кривой в инном представлении точек	15.885.070
оритм MQV на эллиптической кривой в аффинном ставлении точек	31.338.353
Алгоритм Диффи–Хеллмана на эллиптической кривой в проективном представлении точек.	68.745.683
Алгоритм MQV на эллиптической кривой в проективном представлении точек.	105.029.551

Листинг

```
#include <boost/multiprecision/cpp_int.hpp>
#include <boost/random/random_device.hpp>
#include <boost/random.hpp>
#include <boost/multiprecision/integer.hpp>
#include <boost/multiprecision/miller_rabin.hpp>
#include <boost/filesystem/fstream.hpp>
#include <string> #include <iostream>
#include "rdtsc.h"

using namespace std;
using namespace boost::multiprecision; using namespace boost::random;
cpp_int q, p; cpp_int a = -3;

struct proj_rep
{
 cpp_int x; cpp_int y; cpp_int z;
};

proj_rep toProj(pair<cpp_int, cpp_int> a)
{
 return {a.first, a.second, 1};
}

cpp_int bound_random_gen(cpp_int a, cpp_int b)
{
 boost::random::random_device rd; boost::random::mt19937 generator(rd());
 boost::random::uniform_int_distribution<cpp_int> uid(a, b); cpp_int num =
uid(generator);
 return num;
}

bool operator==(proj_rep a, proj_rep b)
{
 return (a.x == b.x && a.y == b.y && a.z == b.z);
}

pair<cpp_int, cpp_int> operator+(pair<cpp_int, cpp_int> A, pair<cpp_int,
cpp_int> B)
{
 cpp_int k = 0;
 pair<cpp_int, cpp_int> RES = {0, 0};
 if (A.first == B.first && A.second == B.second)
       k = (3 * A.first * A.first + a) / (2 * A.second);
 else
       k = (B.second - A.second) / (B.first - A.first); RES.first = (k * k -
 A.first - B.first) % p;
       RES.second = (k * (A.first - RES.first) - B.second) % p; return RES;
}

template <class T> T calculate_composition(cpp_int m, T P)
{
 T Q = P;
 T RES = {-1, -1};
 while (m)
 {
       if (m & 1)
       {
             RES = RES + Q;
       }
       Q = Q + Q; m >>= 1;
 }
 return Q;
}
```

```
proj_rep operator+(proj_rep x, proj_rep y)
{
 proj_rep c; if (x == y)
 {
       cpp_int l1 = (3 * x.x * x.x + a * x.z * x.z * x.z * x.z) % p; cpp_int
       l2 = (4 * x.x * x.y * x.y) % p;
       c.z = (2 * x.y * x.z) % p;
       c.x = (l1 * l1 - 2 * l2) % p;
       cpp_int l3 = (8 * x.y * x.y * x.y * x.y) % p; c.y = (l1 * (l2 - c.x)
 - l3) % p;
 }
 else
 {
       cpp_int l1 = (x.x * y.z * y.z) % p; cpp_int l2 = (y.x * x.z * x.z) %
 p; cpp_int l3 = l2 - l1;
       cpp_int l4 = (x.y * y.z * y.z * y.z) % p; cpp_int l5 = (y.y * x.z *
 x.z * x.z) % p; cpp_int l6 = l5 - l4;
       cpp_int l7 = l1 + l2; cpp_int l8 = l4 + l5;
       c.z = (x.z * y.z * l3) % p;
       c.x = (l6 * l6 - l7 * l3 * l3) % p; cpp_int l9 = l7 * l3 * l3 - 2 *
 c.x;
       c.y = ((l9 * l6 - l8 * l3 * l3 * l3) / 2) % p;
 }
 return c;
}

int main()
{
 long long start = rdtsc();
 const string filename = "parameters.txt"; boost::filesystem::ifstream
input(filename); input >> p >> q;
 input.close(); pair<cpp_int, cpp_int> G =
{(cpp_int)("0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c29
6 "),
(cpp_int)("0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5
"
)};
 proj_rep G1 = toProj(G); cout << 1 << endl;
 cpp_int Xa = bound_random_gen((cpp_int)1, q - 1); cpp_int Xb =
 bound_random_gen((cpp_int)1, q - 1);

 // projective representation
 proj_rep Ya = calculate_composition(Xa, G1); proj_rep Yb =
 calculate_composition(Xb, G1); proj_rep Za = calculate_composition(Xa, Yb);
 proj_rep Zb = calculate_composition(Xb, Ya);
 cout << "Za = " << Za.x << " " << Za.y << " " << Za.z << endl; cout << "Zb
= " << Zb.x << " " << Zb.y << " " << Zb.z << endl;

 // affine representation
 // pair<cpp_int, cpp_int> Ya = calculate_composition(Xa, G);
 // pair<cpp_int, cpp_int> Yb = calculate_composition(Xb, G);
 // pair<cpp_int, cpp_int> Za = calculate_composition(Xa, Yb);
 // pair<cpp_int, cpp_int> Zb = calculate_composition(Xb, Ya);
 // cout << "Za = " << Za.first << " " << Za.second << endl;
 // cout << "Zb = " << Zb.first << " " << Zb.second << endl; long long stop
 = rdtsc();
 cout << "Processor cycles - " << stop - start << endl; return 0;
}
```