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 3](#_Toc105713576)

[Выполнение 4](#_Toc105713577)

[Листинг 10](#_Toc105713578)

# 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:

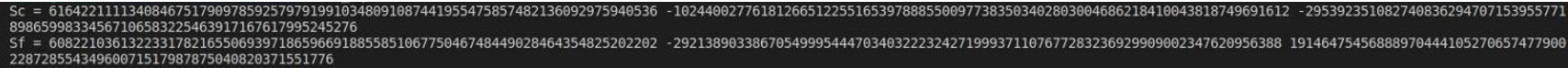


1. В проективном представлении:

Диффи-Хеллман:



MQV:



1. Замеры числа процессорных циклов.

|  |  |
| --- | --- |
|  | Число процессорных циклов |
| Алгоритм Диффи-Хелмана на эллиптической кривой в аффинном представлении точек | 15.885.070 |
| Алгоритм MQV на эллиптической кривой в аффинном представлении 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)("0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296 "),

(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;

}