Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Криптография»**

Студент: Кузьмичев Александр Николаевич

Группа: М80 – 306Б-18

Преподаватель: Борисов А. В.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

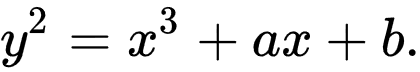
Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Задание:** Подобрать такую эллиптическую кривую над конечным простым полем порядка p, порядок точки которой полным перебором находится за 10 минут на ПК. Упомянуть в отчёте какие алгоритмы и теоремы существуют для облегчения и ускорения решения задачи полного перебора.

**Решение:** Я использовал каноническую форму эллиптической кривой.



Коэффициенты (a, b) я выбрал случайным образом. Модуль кривой (p) пришлось подбирать. Чтобы сориентироваться, я построил график для небольших p, с целью пронаблюдать, как будет изменяться время. После этого, всего за пару итераций я подобрал p = 25013.

**Алгоритм работы:** Находятся все точки, принадлежащие кривой полным перебором. Выбирается случайная точка, находится ее порядок путем сложения самой с собой до тех пор, пока сумма не станет точкой (0, 0). Сложность алгоритма полного перебора – O(p2).

**Листинг кода:**

import time

import random

A = 1860348749492490789823288813930625381760

B = 2001637506671384833171818673149062805974

def curve(x, y, p):

    return (y \*\* 2) % p == (x \*\* 3 + (A % p) \* x + (B % p)) % p

def print\_curve(p):

    print("y^2 = x^3 + {0} \* x + {1} (mod {2})".format(A % p, B % p, p))

def extended\_euclidean\_algorithm(a, b):

    s, prv\_s = 0, 1

    t, prv\_t = 1, 0

    r, prv\_r = b, a

    while r != 0:

        quotient = prv\_r // r

        prv\_r, r = r, prv\_r - quotient \* r

        prv\_s, s = s, prv\_s - quotient \* s

        prv\_t, t = t, prv\_t - quotient \* t

    return prv\_r, prv\_s, prv\_t

def inverse\_of(n, p):

    gcd, x, y = extended\_euclidean\_algorithm(n, p)

    assert (n \* x + p \* y) % p == gcd

    if gcd != 1:

        raise ValueError(

            '{} has no multiplicative inverse '

            'modulo {}'.format(n, p))

    else:

        return x % p

def add\_points(p1, p2, p):

    if p1 == (0, 0):

        return p2

    elif p2 == (0, 0):

        return p1

    elif p1[0] == p2[0] and p1[1] != p2[1]:

        return (0, 0)

    if p1 == p2:

        s = ((3 \* p1[0] \*\* 2 + (A % p)) \* inverse\_of(2 \* p1[1], p)) % p

    else:

        s = ((p1[1] - p2[1]) \* inverse\_of(p1[0] - p2[0], p)) % p

    x = (s \*\* 2 - 2 \* p1[0]) % p

    y = (p1[1] + s \* (x - p1[0])) % p

    return (x, -y % p)

def order\_point(point, p):

    i = 1

    check = add\_points(point, point, p)

    while check != (0, 0):

        check = add\_points(check, point, p)

        i += 1

    return i

if \_\_name\_\_ == '\_\_main\_\_':

    p = 25013

    print\_curve(p)

    points = []

    start = time.time()

    for x in range(0, p):

        for y in range(0, p):

            if curve(x, y, p):

                points.append((x, y))

    cnt\_points = len(points)

    print("Order curve = {0}".format(cnt\_points))

    point = random.choice(points)

    print("Order point {0}: {1}".format(point, order\_point(point, p)))

    print("Time: {} sec.".format(time.time() - start))

**Результаты работы:**

y^2 = x^3 + 12713 \* x + 7802 (mod 25013)

Order curve = 24775

Order point (18056, 24333): 37310

Time: 653.375966458397 sec.

**Выводы:**

Для того, чтобы ускорить процесс вычисления порядка полным перебором можно использовать алгоритм Шуфа. Его сложность – O(log8q). В 1990-х годах американский математик Ноам Элкис предложил улучшенную версию алгоритма. Ожидаемое время работа этого алгоритма – O(log6q). Также существует метод комплексного умножения, с помощью которого можно находить кривые с заданным количеством точек. Но он работает только при выполнении определенных условий.