МИНИСТЕРСТВО НАУКИ И ВЫШСЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №6
по курсу «Криптография»

Студент:	Семин А. В.
Группа:	М8О-306Б-20
Преподаватель:	А. В. Борисов
Оценка:	
Дата:	

Москва, 2023

Лабораторная работа №6

Задание:

Подобрать такую эллиптическую кривую, порядок точки которой полным перебором находится за 10 минут на ПК. Упомянуть в отчёте результаты замеров работы программы, характеристики вычислителя. Также указать какие алгоритмы и/или теоремы существуют для облегчения и ускорения решения задачи полного перебора. Рассмотреть для случая конечного простого поля Z_p.

Ход выполнения работы

Возьмем каноническую формулу эллиптической кривой:

$$y = x^3 + ax + b$$

Коэффициенты **a** и **b** выберем случайным образом.

Модуль кривой **p** подберем вручную, исходя из условий, что это должно быть простое число и полной перебор должен работать около десяти минут. Имеем ввиду, что приблизительное время работы перебора для квадрата чисел порядка 10^4. Мои вычислительны характеристики: процессор AMD Ryzen 7 4800H 2.90 GHz, 16 гб ОЗУ DDR4. Исходя из всего этого, возьмем приблизительное число p = 30000.

Поиск всех точек – процесс продолжительный, асимптотическая сложность алгоритма полного перебора $O(p^2)$. Далее ищем порядок точки, случайно выбранной из найденных. Складываем ее с ней самой же, пока не получим нулевую точку. Количество операций сложения и будет являться искомым порядком точки. А число точек, которые принадлежат кривой, - порядок кривой.

```
import random
import time
import matplotlib.pyplot as plt
import numpy as np

A = random.randint(1000000000, 10000000000)
B = random.randint(1000000000, 10000000000)

def elliptic_curve(x, y, p):
    return (y ** 2) % p == (x ** 3 + (A % p) * x + (B % p)) % p

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


def find_points():
    points = []
    for x in range(p):
        for y in range(p):
            if elliptic_curve(x, y, p):
                points.append((x, y))
    return points

def extended_euclidean_algorithm(a, b):
    s, old_s = 0, 1
    t, old_t = 1, 0
```

```python
    r, old_r = b, a
    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_s, s = s, old_s - quotient * s
        old_t, t = t, old_t - quotient * t
    return old_r, old_s, old_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):
    x1, y1 = p1[0], p1[1]
    x2, y2 = p2[0], p2[1]
    if p1 == (0, 0):
        return p2
    elif p2 == (0, 0):
        return p1
    elif x1 == x2 and y1 != y2:
        return (0, 0)

    if p1 == p2:
        m = ((3 * x1 ** 2 + (A % p)) * inverse_of(2 * y1, p)) % p
    else:
        m = ((y1 - y2) * inverse_of(x1 - x2, p)) % p

    x3 = (m ** 2 - x1 - x2) % p
    y3 = (y1 + m * (x3 - x1)) % p
    return [x3, -y3 % p]


def point_order(point, p):
    i = 1
```

```
    new_point = add_points(point, point, p)
    while new_point != (0, 0):
        new_point = add_points(new_point, point, p)
        i += 1
    return i

def sieve(n):
    primes = 2 * [False] + (n - 1) * [True]
    for i in range(2, int(n ** 0.5 + 1.5)):
        for j in range(i * i, n + 1, i):
            primes[j] = False
    return [prime for prime, checked in enumerate(primes) if checked]

if __name__ == '__main__':
    primes = sieve(30000)
    p = primes[-1]
    start = time.time()
    points = find_points()
    points_num = len(points)
    print_curve()
    print("Порядок кривой = {0}".format(points_num))
    point = random.choice(points)
    print("Порядок точки {0}: {1}".format(point, point_order(point, p)))
    print("Время: {0}".format(time.time() - start))
```

Вывод в консоль:

```
y^2 = x^3 + 14593 * x + 29778 (mod 29989)
Порядок кривой = 30123
Порядок точки (15362, 26940): 1771
Время: 598.070111989975
```

Вывод

В ходе выполнения данной лабораторной работы мы поработали с эллиптическими кривыми и подсчитали число точек на эллиптической кривой. Эллиптические кривые используются в криптографии для генерации ключей, потому что для поиска параметра, на которое производится умножение, требуется решить задачу дискретного логарифмирования на эллиптической кривой. Зная начальное и конечное значения, найти этот параметр не представляется возможным за приемлемое время. Полный перебор работает слишком долго, но есть алгоритмы для ускорения перебора точек. Например, алгоритм Шуфа, который в своей основе использует теорему Хассе и имеет асимптотическую сложность $\mathbf{O(\log^8 q)}$ (q – число элементов поля).