**Отчет по лабораторной работе №6 по курсу «Криптография»**

Выполнил Лисин Роман Сергеевич, М8О-306Б-20.

***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.

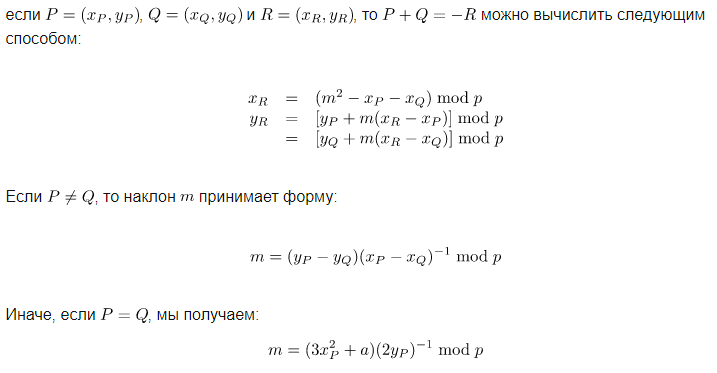
***Ход работы***

Рассмотрим эллиптическую кривую , где в конечном простом поле . Коэффициенты a = 0 и b = 7 я задал сам из головы.

Для нахождения порядка эллиптической кривой я нахожу количество целочисленных точек из множества , принадлежащих заданной кривой. Эта операция выполняется за .

Затем я выбираю случайную точку и начинаю искать её порядок. Для этого я складываю точку саму с собой до тех пор, пока не получится точка (0, 0). Количество итераций, потребовавшихся на это, и есть порядок точки. Все вычисления проводятся по модулю p. Алгебраическое сложение двух точек в

проводится по следующим правилам:



(взято с <https://habr.com/ru/post/335906/> )

Порядок поля p = 31277 был подобран экспериментально. Я брал простые числа и смотрел на время, которое требовалось для нахождения порядка точки в соответствующем поле. Результат - примерно 598 секунд, что примерно 10 минут.

***Код***

import time  
import random  
  
from tqdm import tqdm # Использую этот модуль, чтобы примерно понимать, сколько времени будет работать программа.  
# Он рисует progress bar.  
  
  
def elliptic\_curve(x, y, p):  
 *"""*  
 *Проверяем, что точка (x, y) принадлежит эллиптической кривой y^2 = x^3 + ax + b в поле Z\_p*  
 *"""*  
return (y \*\* 2) % p == (x \*\* 3 + (a % p) \* x + (b % p)) % p  
  
  
def extended\_euclidean\_algorithm(a, b):  
 *"""*  
 *Возвращает (gcd, x, y): ax + by == gcd(a, b)*  
 *Сложность: O(log b)*  
 *Взято с https://habr.com/ru/post/335906/*  
 *"""*  
s, old\_s = 0, 1  
 t, old\_t = 1, 0  
 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):  
 *"""*  
 *Возвращает m: (n \* m) % p == 1*  
 *Взято с https://habr.com/ru/post/335906/*  
 *"""*  
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 points\_sum(A, B, p):  
 *"""*  
 *Вычисляем алгебраическую сумму точек A, B в поле Z\_p*  
 *Алгоритм с https://habr.com/ru/post/335906/*  
 *Возвращает R = (x\_r, y\_r) = A + B*  
 *"""*  
if A == (0, 0):  
 return B  
 if B == (0, 0):  
 return A  
 if A[0] == B[0] and A[1] != B[1]:  
 return 0, 0  
  
 if A != B:  
 m = ((A[1] - B[1]) \* inverse\_of(A[0] - B[0], p)) % p  
 else:  
 m = ((3 \* A[0] \*\* 2 + a) \* inverse\_of(2 \* A[1], p)) % p  
  
 x\_r = (m \*\* 2 - A[0] - A[1]) % p  
 y\_r = (A[1] + m \* (x\_r - A[0])) % p  
 return x\_r, -y\_r % p  
  
  
def get\_point\_order(point, p):  
 *"""*  
 *Вычисляем порядок точки point в поле Z\_p*  
 *"""*  
ans = 0  
 found\_point\_order = False  
 prev\_point = point  
 while not found\_point\_order:  
 ans += 1  
 point\_sum = points\_sum(point, prev\_point, p)  
 if point\_sum == (0, 0):  
 found\_point\_order = True  
 else:  
 prev\_point = point  
 point = point\_sum  
 return ans  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 a = 0  
 b = 7  
 p = 31277  
  
 start = time.time()  
 points = []  
 for x in tqdm(range(p)):  
 for y in range(p):  
 if elliptic\_curve(x, y, p):  
 points.append((x, y))  
  
 curve\_order = len(points)  
 print('Curve order:', curve\_order)  
  
 point = random.choice(points)  
 point\_order = get\_point\_order(point, p)  
 print('Point', point, 'order:', point\_order)  
  
 end = time.time()  
 print('Time:', end - start, 's')

***Результат***

100%|██████████| 31277/31277 [09:57<00:00, 52.36it/s]  
Curve order: 31277  
Point (12652, 1851) order: 96397  
Time: 597.6167588233948 s

Вычисления производились на процессоре AMD Ryzen 7 5700u with radeon graphics × 16.

Также хочется отметить, что полный перебор для поиска порядка кривой можно заменить на алгоритм Шуфа или алгоритм Шуфа-Элкиса-Аткина, которые работают за полиномиальное время. В этих алгоритмах используется теорема Хассе.