**Отчет по лабораторной работе №3 по курсу «Криптография»**

Выполнил Моисеенков Илья Павлович, М8О-308Б-19.

***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 .

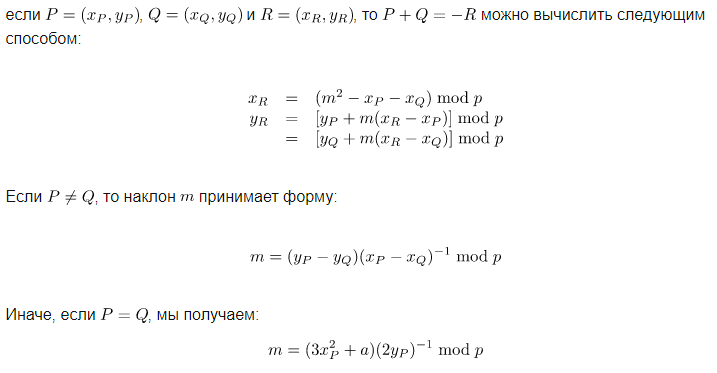
***Ход работы***

Рассмотрим эллиптическую кривую , где в конечном простом поле . Коэффициенты a и b я задал рандомно.

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

Затем я выбираю рандомную точку и начинаю искать ее 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/> )

Будем рассматривать кривую *2683* в поле . Порядок поля был подобран экспериментально. Я брал разные простые числа и смотрел на время, которое требовалось для нахождения порядка точки в соответствующем поле.

***Код***

import time  
import random  
a = 2683  
b = 2399  
  
  
def elliptic\_curve(x, y, p):  
 """  
 Check if point (x, y) is in elliptic curve y^2 = x^3 + ax + b in Z\_p  
 Returns true or false  
 """  
 return (y \*\* 2) % p == (x \*\* 3 + (a % p) \* x + (b % p)) % p  
  
  
def extended\_euclidean\_algorithm(a, b):  
 """  
 Returns (gcd, x, y): ax + by == gcd(a, b)  
 Complexity: O(log b)  
 stolen from 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):  
 """  
 Returns m: (n \* m) % p == 1  
 stolen from 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):  
 """  
 Get algebraic sum of two points A, B in Z\_p  
 Algorithm: https://habr.com/ru/post/335906/  
 Returns 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):  
 """  
 Get order of the point in 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\_\_':  
 p = 32003  
  
 start = time.time()  
 points = []  
 for x in 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***

Curve order: 31986

Point (30080, 14559) order: 59998

Time: 856.1665139198303 s

Вычисления производились на процессоре Intel Core i5-8250U.

***Выводы***

Выполняя эту работу, я познакомился с криптографией на эллиптических кривых. Эллиптическая кривая — это кривая вида , заданная на поле . С виду кажется, что это уравнение совсем не похоже на что-то, что может использоваться в шифровании. Но это не так.

С помощью эллиптических кривых может производиться шифрование и передача сообщений с использованием открытых/закрытых ключей. Через эллиптические кривые можно реализовать электронные подписи. Используя эллиптические кривые, можно факторизовать большие числа (я уже сталкивался с таким алгоритмом в прошлой лабораторной).

В этой лабораторке я смог сам подобрать такую эллиптическую кривую, поиск порядка точек которой занимает довольно большое время. Я разобрался в деталях, как вычисляется порядок кривой и порядок ее точек. Но я вычислял порядок кривой за квадрат - простым перебором. Этот этап можно упростить, используя алгоритм Шуфа, который работает за полиномиальное время.