Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу «Криптография»

Студент: А. П. Шорохов
Преподаватель: А. В. Борисов
Группа: М8О-306Б
Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №4

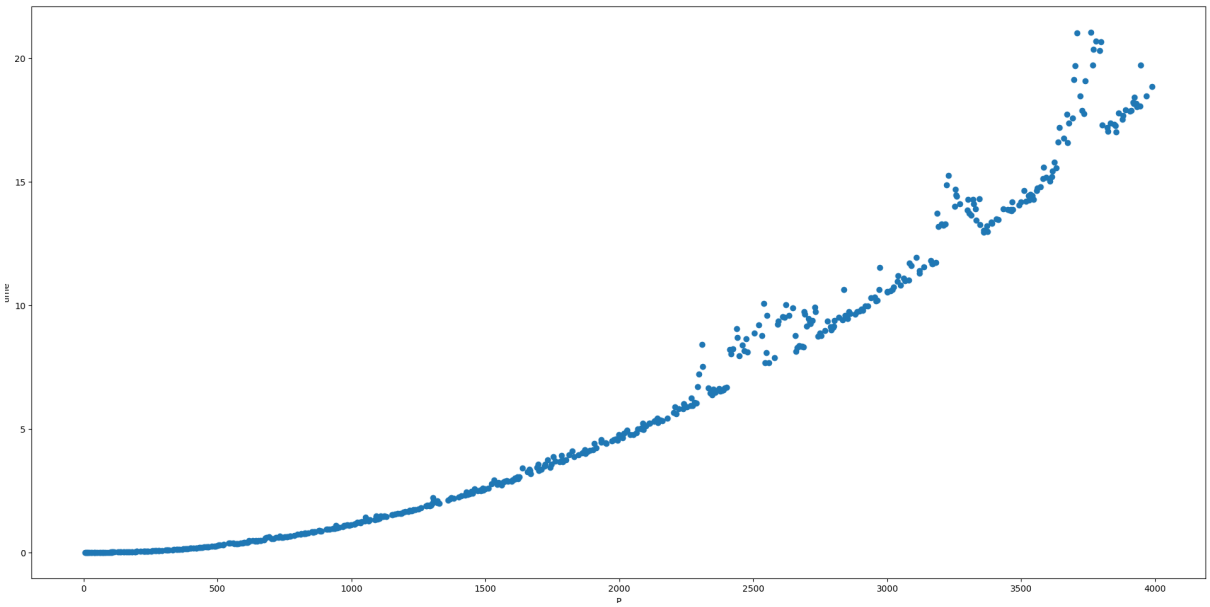
Задача: Подобрать такую эллиптическую кривую над конечным простым полем порядка p, такую, порядок точки которой полным перебором находится за 10 минут на ПК. Упомянуть в отчёте какие алгоритмы и теоремы существуют для облегчения и ускорения решения задачи полного перебора.

1 Метод решения

Каноническая форма эллиптической кривой : $y^2 = x^3 + ax + b$
Коэффициенты a и b выберем рандомно.

Алгоритм работы программы:
Полным перебором за $O(p^2)$ находим все точки кривой. Берём случайную точку, находим её порядок путём сложения с самой собой до тех пор, пока сумма не станет точкой (0, 0). Количество операций сложения будет искомым порядком.

Для подбора подходящего p построим график зависимости затраченного времени от величины p. Получилось вот такое распределение.



Для получения этих данных мы прошли по всем простым p до 4000.

Имея представление о примерном времени, запустим программу на массиве p = [15000, 20000, 25000, 30000, 35000].
Получим следующее время:

```
dobriy_alien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/3course/crypto/lab4$
python3 main.py
P suggestion = 15000
y^2 = x^3 + 12175 * x + 10718 (mod 14983)
```

```
Elliptic curve order = 15091
Point order (4700,10871): 7545
Time: 176.9085237979889

P suggestion = 20000
y^2 = x^3 + 3565 * x + 10799 (mod 19997)
Elliptic curve order = 20185
Point order (10215,7214): 20185
Time: 316.91879987716675

P suggestion = 25000
y^2 = x^3 + 2970 * x + 5816 (mod 24989)
Elliptic curve order = 25209
Point order (9757,17680): 12604
Time: 499.7667100429535

P suggestion = 30000
y^2 = x^3 + 861 * x + 21110 (mod 29989)
Elliptic curve order = 30123
Point order (18098,13472): 30123
Time: 719.6785411834717

P suggestion = 35000
y^2 = x^3 + 173 * x + 31922 (mod 34981)
Elliptic curve order = 34945
Point order (29336,8195): 17472
Time: 975.3128824234009
```

Видим, что искомый p располагается между 25000 и 30000. Поэтому возьмем $p = 27500$.
В результате имеем:

```
dobriy_alien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/3course/crypto/lab4$
python3 main.py
P suggestion = 27500
y^2 = x^3 + 5308 * x + 9130 (mod 27487)
Elliptic curve order = 27596
Point order (20187,10726): 27596
Time: 623.6328580379486
```

2 Исходный код

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

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

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

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

def extendedEuclideanAlgorithm(a, b):
    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(n, p):
    gcd, x, y = extendedEuclideanAlgorithm(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 addPoints(p1, p2, p):
    x1, y1 = p1[0], p1[1]
    x2, y2 = p2[0], p2[1]
```

```
48
49      if p1 == (0, 0):
50          return p2
51      elif p2 == (0, 0):
52          return p1
53      elif x1 == x2 and y1 != y2:
54          return (0, 0)
55
56
57
58      if p1 == p2:
59          m = ((3 * x1 ** 2 + (A % p)) * inverse(2 * y1, p)) % p
60      else:
61          m = ((y1 - y2) * inverse(x1 - x2, p)) % p
62
63
64      x3 = (m ** 2 - x1 - x2) % p
65      y3 = (y1 + m * (x3 - x1)) % p
66
67      return [x3, -y3 % p]
68
69
70  def pointOrder(point, p):
71      i = 1
72      new_point = addPoints(point, point, p)
73      while new_point != (0, 0):
74          new_point = addPoints(new_point, point, p)
75          i += 1
76
77      return i
78
79  def sieve(n):
80      primes = 2 * [False] + (n - 1) * [True]
81      for i in range(2, int(n ** 0.5 + 1.5)):
82          for j in range(i * i, n + 1, i):
83              primes[j] = False
84      return [prime for prime, checked in enumerate(primes) if checked]
85
86  def printIt(p):
87      print("y^2 = x^3 + {0} * x + {1} (mod {2})".format(A % p, B % p, p))
88
89  def run(sugg):
90      primes = sieve(sugg)
91      p = primes[-1]
92
93      start = time.time()
94
95      points = searchPoints(p)
96
```

```
97      points_num = len(points)
98
99      printIt(p)
100     print("Elliptic curve order = {0}".format(points_num))
101
102     point = random.choice(points)
103
104     print("Point order {0}: {1}".format(point, pointOrder(point, p)))
105     print("Time: {0}".format(time.time() - start))
106
107
108 if __name__ == '__main__':
109
110     #suggest = [15000, 20000, 25000, 30000, 35000] # to find the right p
111     suggest = [27500]
112
113     for i in suggest:
114         print('P suggestion = {0}'.format(i))
115         run(i)
116         print()
```

3 Выводы

Для ускорения решения задачи полного перебора можно использовать такие алгоритмы, как алгоритм Шуфа, использующий теорему Хассе, работающий за $O(log^8 q)$, или метод комплексного умножения, который позволяет более эффективно находить кривые с заданным количеством точек.