

# Отчёт по лабораторной работе 4

Студент: Кочкожаров Иван Вячеславович

Группа: М8О-308Б-22

## 1 Цель работы

Подобрать такую эллиптическую кривую, порядок точки которой полным перебором находится за 10 минут на ПК. Исследовать алгоритмы и теоремы, существующие для облегчения и ускорения решения задачи полного перебора.

## 2 Теоретическая часть

### 2.1 Эллиптические кривые над конечным полем

Эллиптическая кривая над конечным полем  $\mathbb{F}_p$  (где  $p$  — простое число) определяется уравнением:

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad (1)$$

При условии, что  $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$  (условие несингулярности).

Точки эллиптической кривой образуют абелеву группу с операцией сложения, определенной геометрически. Порядок точки  $P$  — это наименьшее положительное целое число  $n$  такое, что  $nP = \mathcal{O}$  (бесконечно удаленная точка).

### 2.2 Оценка производительности

Для оценки среднего времени выполнения одной операции сложения точек был проведен эксперимент с кривой над полем  $\mathbb{F}_p$  при  $p = 50\,000\,017$ :

```
1 p=50000017, a=6470481, b=8975775, P=(12195349,21713483), order
  =49992337, time=100.81s
2 Avg iter time = 2.0164165265300723e-06
```

Среднее время одной операции сложения составило примерно  $2.02 \cdot 10^{-6}$  с.

### 2.3 Оценка времени вычисления порядка точки

Если каждая операция сложения занимает примерно  $2 \cdot 10^{-6}$  с ( $2 \mu\text{s}$ ), то за 10 минут (600 с) вы сможете сделать порядка

$$\frac{600 \text{ с}}{2 \cdot 10^{-6} \text{ с/операцию}} = 3 \cdot 10^8 \text{ скалярных сложений.}$$

Поскольку порядок точки на «случайной» кривой над  $\mathbb{F}_p$  примерно равен  $p$  (в пределах  $\pm\sqrt{p}$ ) по теореме Хассе, чтобы перебор  $kP$  для  $k = 1 \dots p$  занял около 600 с, достаточно взять

$$p \approx 3 \cdot 10^8.$$

Например, одно из ближайших простых чисел порядка  $6 \cdot 10^8$  —

$$p = 300\,000\,007.$$

Его используем для начального приближения кривой.

## 3 Практическая часть

### 3.1 Реализация алгоритмов

Для решения задачи были реализованы следующие компоненты:

1. Класс `EllipticCurve` для представления эллиптической кривой
2. Класс `Point` для представления точек на кривой
3. Функции для операций над точками: `add` и `mul`
4. Функция `brute_order` для вычисления порядка точки методом полного перебора
5. Функции для генерации случайных кривых и точек

### 3.2 Подбор параметров кривой

Для поиска подходящей кривой был реализован алгоритм, который:

1. Генерирует случайную эллиптическую кривую над полем  $\mathbb{F}_p$
2. Генерирует случайную точку на этой кривой
3. Вычисляет порядок точки методом полного перебора
4. Проверяет, что время вычисления находится в заданном диапазоне (9-11 минут)

Были проведены эксперименты с различными значениями  $p$ :

```
1 p=300000007, a=139493976, b=266004513, P=(101097864,85148484), order
  =300002521, time=440.46s
2 Too fast: only 440.46s
3 p=300000007, a=127506766, b=130653405, P=(125936178,88420890), order
  =150015410, time=225.28s
4 Too fast: only 225.28s
5 p=300000007, a=41635840, b=112170592, P=(107361594,131765451), order
  =300010261, time=445.85s
6 Too fast: only 445.85s
7 p=300000007, a=177004968, b=43109284, P=(184984415,9153220), order
  =300000181, time=446.48s
8 Too fast: only 446.48s
```

```
1 p=426000017, a=64982240, b=19135670, P=(228072906,97062984), order
  =213018587, time=332.94s
2 Too fast: only 332.94s
3 p=426000017, a=131325779, b=8713283, P=(230519031,147449158), order
  =425998575, time=658.79s
4
5 Selected curve and point:
```

```

6 y^2 = x^3 + 131325779x + 8713283 (mod 426000017)
7 Point P = (230519031, 147449158), order = 425998575, computed in 658.79
  s

1 p=387272759, a=331128336, b=169430433, P=(164198548,22675309), order
  =129094316, time=204.41s
2 Too fast: only 204.41s
3 p=387272759, a=145203106, b=273166277, P=(116919177,167275377), order
  =96815321, time=155.08s
4 Too fast: only 155.08s
5 p=387272759, a=124860295, b=2186117, P=(223741075,161701677), order
  =387264450, time=621.83s
6
7 Selected curve and point:
8 y^2 = x^3 + 124860295x + 2186117 (mod 387272759)
9 Point P = (223741075, 161701677), order = 387264450, computed in 621.83
  s

```

- $p = 300\,000\,007$  (результаты оказались слишком быстрыми, около 440 с)
- $p = 426\,000\,017$  (найдена кривая с временем вычисления 658.79 с)
- $p = 387\,272\,759$  (найдена кривая с временем вычисления 621.83 с)

## 4 Результаты

В результате экспериментов была найдена оптимальная эллиптическая кривая:

$$y^2 = x^3 + 124\,860\,295x + 2\,186\,117 \pmod{387\,272\,759} \quad (2)$$

$$P = (223\,741\,075, 161\,701\,677) \quad (3)$$

$$\text{Порядок точки } P = 387\,264\,450 \quad (4)$$

$$\text{Время вычисления} = 621.83 \text{ с} \quad (5)$$

Это значение находится в пределах заданного диапазона (9-11 минут) и подтверждает теоретические оценки.

## 5 Алгоритмы для ускорения вычисления порядка точки

Существуют более эффективные алгоритмы для вычисления порядка точки на эллиптической кривой:

### 5.1 Алгоритм Шенкса (Baby-step Giant-step)

Алгоритм Шенкса позволяет найти порядок точки за время  $O(\sqrt{n})$ , где  $n$  — порядок группы. Основная идея:

1. Выбрать параметр  $m \approx \sqrt{n}$
2. Вычислить и сохранить точки  $jP$  для  $j = 0, 1, \dots, m-1$
3. Вычислить точки  $P_0 - imP$  для  $i = 0, 1, \dots, m$ , где  $P_0$  — некоторая известная точка
4. Найти совпадение между двумя наборами точек

## 5.2 Алгоритм Полларда $\rho$

Алгоритм Полларда использует псевдослучайную функцию для поиска цикла в последовательности точек. Время работы также  $O(\sqrt{n})$ , но требует меньше памяти, чем алгоритм Шенкса.

## 5.3 Теорема Хассе

Теорема Хассе утверждает, что порядок эллиптической кривой  $E$  над полем  $\mathbb{F}_p$  находится в диапазоне:

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p} \quad (6)$$

Это позволяет существенно сузить область поиска порядка кривой.

## 5.4 Алгоритм Шуфа

Алгоритм Шуфа позволяет вычислить точное значение порядка эллиптической кривой за полиномиальное время. Он основан на вычислении порядка кривой по модулю малых простых чисел и последующем применении Китайской теоремы об остатках.

## 6 Характеристики вычислителя

- Процессор: AMD Ryzen 5 5600H (8 ядер, 16 потоков)
- Оперативная память: 16 ГБ DDR4 3200 МГц
- Операционная система: Arch Linux (ядро 6.14.6-arch1-1)
- Python: версия 3.13.3

## 7 Выводы

1. Была подобрана эллиптическая кривая, для которой вычисление порядка точки методом полного перебора занимает около 10 минут (621.83 с).
2. Экспериментально подтверждены теоретические оценки времени вычисления.
3. Рассмотрены более эффективные алгоритмы для вычисления порядка точки, которые могут значительно ускорить процесс по сравнению с полным перебором.
4. Полученные результаты демонстрируют важность использования оптимизированных алгоритмов при работе с эллиптическими кривыми в криптографических приложениях.

## 8 Приложение: исходный код

<https://github.com/kochkozharov/cryptography-labs/tree/master/lab4>

## 8.1 es.py - Реализация эллиптических кривых

```
1 import random
2 import signal
3 import time
4
5 class EllipticCurve:
6     def __init__(self, a, b, p):
7         self.a = a
8         self.b = b
9         self.p = p
10        if (4 * a**3 + 27 * b**2) % p == 0:
11            raise ValueError("Singular curve")
12
13 class Point:
14     def __init__(self, x=None, y=None, curve=None):
15         self.x = x
16         self.y = y
17         self.curve = curve
18
19     def is_infinity(self):
20         return self.x is None and self.y is None
21
22     @staticmethod
23     def infinity(curve):
24         return Point(None, None, curve)
25
26 def add(P, Q):
27     curve = P.curve
28     p = curve.p
29     if P.is_infinity(): return Q
30     if Q.is_infinity(): return P
31     if P.x == Q.x and (P.y + Q.y) % p == 0:
32         return Point.infinity(curve)
33     if P.x == Q.x:
34         lam = (3 * P.x * P.x + curve.a) * pow(2 * P.y, -1, p) % p
35     else:
36         lam = ((Q.y - P.y) * pow(Q.x - P.x, -1, p)) % p
37     x_r = (lam * lam - P.x - Q.x) % p
38     y_r = (lam * (P.x - x_r) - P.y) % p
39     return Point(x_r, y_r, curve)
40
41 def mul(P, n):
42     R = Point.infinity(P.curve)
43     Q = P
44     while n > 0:
45         if n & 1:
46             R = add(R, Q)
47             Q = add(Q, Q)
48             n >>= 1
49     return R
50
51 def get_random_xy(E):
```

```

52     while True:
53         x = random.randrange(E.p)
54         rhs = (x**3 + E.a*x + E.b) % E.p
55         y = None
56         for yy in range(E.p):
57             if yy*yy % E.p == rhs:
58                 y = yy
59                 break
60         if y:
61             break
62         else:
63             continue
64     return x, y
65
66 def get_random_curve(p):
67     while True:
68         a = random.randrange(p)
69         b = random.randrange(p)
70         try:
71             E = EllipticCurve(a, b, p)
72             break
73         except ValueError:
74             continue
75     return E
76
77 class TimeoutError(Exception):
78     pass
79
80 def _timeout_handler(signum, frame):
81     raise TimeoutError
82
83 def brute_order(P, max_seconds=660):
84     signal.signal(signal.SIGALRM, _timeout_handler)
85     signal.alarm(max_seconds)
86     start = time.perf_counter()
87     try:
88         R = Point.infinity(P.curve)
89         n = 1
90         while True:
91             R = add(R, P)
92             if R.is_infinity():
93                 break
94             n += 1
95         elapsed = time.perf_counter() - start
96         signal.alarm(0)
97         return n, elapsed
98     except TimeoutError:
99         elapsed = time.perf_counter() - start
100         signal.alarm(0)
101         return None, elapsed

```

## 8.2 main.py - Поиск подходящей кривой

```

1 from ec import *
2
3
4 def find_curve(p, min_seconds=540, max_seconds=660):
5     while True:
6         E = get_random_curve(p)
7         x, y = get_random_xy(E)
8         P = Point(x, y, E)
9         order, elapsed = brute_order(P, max_seconds)
10        print(f"p={p}, a={E.a}, b={E.b}, P=({x},{y}), order={order},
11              time={elapsed:.2f}s")
12        if order is None:
13            print(f"Timeout after {max_seconds}s")
14            continue
15        if elapsed < min_seconds:
16            print(f"Too fast: only {elapsed:.2f}s")
17            continue
18        return E, P, order, elapsed
19
20 if __name__ == '__main__':
21     p = 387272759
22     curve, point, order, elapsed = find_curve(p)
23     print("\nSelected curve and point:")
24     print(f"y^2 = x^3 + {curve.a}x + {curve.b} (mod {curve.p})")
25     print(f"Point P = ({point.x}, {point.y}), order = {order}, computed
26           in {elapsed:.2f}s")

```

### 8.3 estimation.py - Измерение среднего времени операции

```

1 from ec import *
2 import time
3
4 if __name__ == '__main__':
5     p = 50000017
6     E = get_random_curve(p)
7     x, y = get_random_xy(E)
8     P = Point(x, y, E)
9     time_sum=0
10    R = Point.infinity(P.curve)
11    n = 1
12    while True:
13        start = time.perf_counter()
14        R = add(R, P)
15        if R.is_infinity():
16            break
17        n += 1
18        elapsed = time.perf_counter() - start
19        time_sum+=elapsed
20    print(f"p={p}, a={E.a}, b={E.b}, P=({x},{y}), order={n}, time={
21          time_sum:.2f}s")
22    print(f"Avg iter time = {time_sum / (n-1)}")

```