**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО 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 № 5  
по курсу «Криптография»

Группа: М8О-308Б-21

Студент(ка): К.А.Белоносов

Преподаватель: А. В. Борисов

Оценка:

Дата: 17.05.2024

Москва, 2024

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc158983147)

[2 Задание 3](#_Toc158983148)

[3 Теория 4](#_Toc158983149)

[4 Ход лабораторной работы 6](#_Toc158983150)

[5 Выводы 10](#_Toc158983151)

# **Тема**

Криптография на эллиптических кривых

# **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.

# **Теория**

**Криптография на эллиптических кривых (ECC)** является важной областью современной криптографии, которая основывается на свойствах эллиптических кривых. Она используется для создания криптографических алгоритмов, обеспечивающих высокий уровень безопасности при меньших размерах ключей по сравнению с традиционными методами, такими как RSA.

Эллиптическая кривая в криптографии представляет собой набор точек, удовлетворяющих уравнению вида:

𝑦2=𝑥3+𝑎𝑥+𝑏y2=x3+ax+b

где 𝑎 и 𝑏 — некоторые константы, определяющие форму кривой. Эти кривые обладают интересными алгебраическими свойствами, которые делают их полезными для криптографии.

Основное свойство эллиптических кривых, используемое в криптографии, заключается в возможности определения операции сложения точек на кривой. Если у нас есть две точки 𝑃 и 𝑄 на эллиптической кривой, то их сумма 𝑅=𝑃+𝑄 также будет точкой на этой кривой. Эта операция удовлетворяет определенным математическим свойствам, таким как ассоциативность и коммутативность.

Арифметика на эллиптических кривых включает в себя следующие основные операции:

1. Сложение точек: Для двух точек 𝑃(𝑥1,𝑦1) 𝑄(𝑥2,𝑦2) их сумма 𝑅(𝑥3,𝑦3) определяется следующим образом:

* Если 𝑃≠𝑄P

𝜆=𝑦2−𝑦1𝑥2−𝑥1

𝑥3=𝜆2−𝑥1−𝑥2

𝑦3=𝜆(𝑥1−𝑥3)−𝑦1

* Если 𝑃=𝑄 (удвоение точки):

𝜆=3𝑥12+𝑎2𝑦1

𝑥3=𝜆2−2𝑥1

𝑦3=𝜆(𝑥1−𝑥3)−𝑦1

1. Скалярное умножение: Это операция многократного сложения точки 𝑃 с самой собой. Если 𝑘 — скаляр, то 𝑘𝑃 обозначает точку, полученную путем сложения 𝑃 с самой собой 𝑘 раз.

**Дискретный случай эллиптических кривых на 𝑍/𝑝**

В криптографии часто рассматриваются эллиптические кривые над конечными полями, например, 𝑍/𝑝, где 𝑝 — простое число. В этом случае уравнение эллиптической кривой принимает вид:

𝑦2≡𝑥3+𝑎𝑥+𝑏 (mod 𝑝)

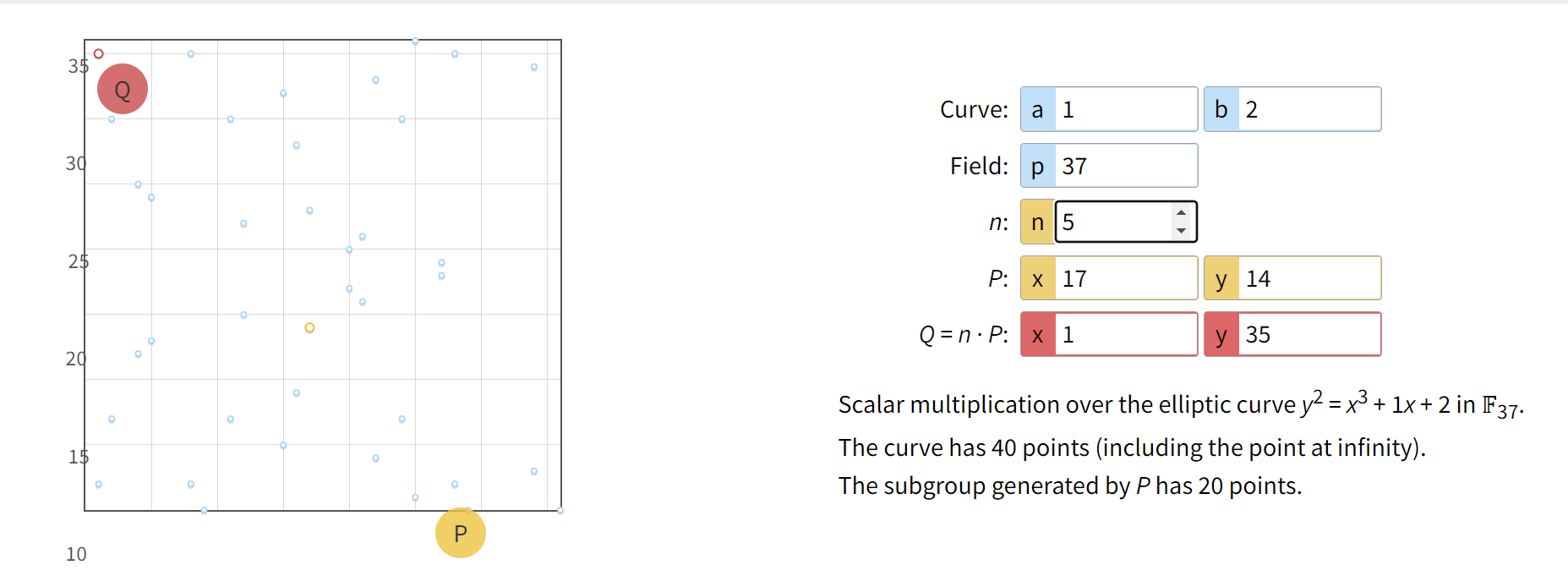
Здесь арифметика ведется по модулю 𝑝p. Это означает, что все вычисления с координатами точек происходят в поле вычетов по модулю 𝑝p. В таком контексте добавление и умножение точек выполняются с использованием операций по модулю 𝑝p.

Эллиптические кривые над конечными полями обладают рядом преимуществ для криптографии:

* Эффективность: Они позволяют достичь высокой безопасности при меньших размерах ключей по сравнению с традиционными системами (например, RSA).
* Безопасность: Сложность вычисления дискретного логарифма на эллиптической кривой делает такие системы устойчивыми к криптоаналитическим атакам.

# **Ход лабораторной работы**

Для выполнения данной лабораторной работы я познакомился со статьей преподавателя посвященной эллиптическим кривым. И узнал основы теории. В статье прикладывался сайт, на котором можно поработать с калькулятором арифметики эллиптических кривых на R и на конечном поле Z/p.



Данный сайт помог в отладке программы. Чтобы узнать порядок точки, следует складывать её саму с собой до тех пор, пока она не обнулиться. В статье прикладывались формулы для сложения. Я написал программу на языке python и протестировал корректность её работы с помощью калькулятора. Решать задачу нахождения модуля я использовал кривую **brainpoolP160r1** со следующими параметрами:

a = 2010641399982644954973368651168534803387298831710787784499056

b = 109831868168684366309758963308953137708728952

Вывод программы:  
**p: 100000007, Point: (0, 21962169), Order: 100007841**

**Elapsed Time: 607.88 seconds**

Как можно заметить при росте модуля сложность алгоритма сильно возрастает.

import time

from sympy import mod\_inverse, isprime, nextprime

import matplotlib.pyplot as plt

class EllipticCurve:

def \_\_init\_\_(self, a, b, p):

self.a = a

self.b = b

self.p = p

def is\_on\_curve(self, x, y):

return (y \* y) % self.p == (x \* x \* x + self.a \* x + self.b) % self.p

def add\_points(self, P, Q):

if P == (None, None):

return Q

if Q == (None, None):

return P

x1, y1 = P

x2, y2 = Q

if x1 == x2 and y1 != y2:

return (None, None)

if x1 == x2:

m = (3 \* x1 \* x1 + self.a) \* mod\_inverse(2 \* y1, self.p)

else:

m = (y2 - y1) \* mod\_inverse(x2 - x1, self.p)

m = m % self.p

x3 = (m \* m - x1 - x2) % self.p

y3 = (m \* (x1 - x3) - y1) % self.p

return (x3, y3)

def find\_order(E, P):

n = 1

Q = P

while Q != (None, None):

Q = E.add\_points(Q, P)

n += 1

return n

def main():

a = 2010641399982644954973368651168534803387298831710787784499056

b = 109831868168684366309758963308953137708728952

p = 10000000

times = []

ps = []

while True:

while not isprime(p):

p = nextprime(p)

E = EllipticCurve(a, b, p)

found = False

for x in range(p):

for y in range(p):

if E.is\_on\_curve(x, y) and y != 0:

P = (x, y)

start\_time = time.time()

order = find\_order(E, P)

if order > 1:

elapsed\_time = time.time() - start\_time

times.append(elapsed\_time)

ps.append(p)

print(f"p: {p}, Point: {P}, Order: {order}")

print(f"Elapsed Time: {elapsed\_time:.2f} seconds")

found = True

break

if found:

break

elapsed\_time = time.time() - start\_time

if elapsed\_time > 600:

print(f"Elapsed time exceeded 10 minutes. Stopping. Last p: {p}, Elapsed Time: {elapsed\_time:.2f} seconds")

break

p = nextprime(p)

if \_\_name\_\_ == "\_\_main\_\_":

main()

Существует еще несколько алгоритмов, которые позволяют ускорить. Алгоритм Полларда ρ, алгоритм baby-step giant-step. Оба алгоритма имеют временную сложность . Различие в том, что алгоритм Полларда требует O(1) памяти, что делает его более применимым и в целом он требует меньшее число шагов, но время работы получается больше, чем у алгоритма baby-step giant-step

# **Выводы**

В результате данной лабораторной работы я познакомился с процессом нахождения порядка точки на эллиптической кривой над конечным полем. Проведенные вычисления показали, что при использовании наивного алгоритма время выполнения значительно увеличивается с ростом модуля, что делает данный подход непрактичным для больших значений.

Я изучил более эффективные алгоритмы для нахождения порядка точки, такие как алгоритм Полларда ρ, алгоритм baby-step giant-step, которые позволяют значительно ускорить вычисления. Кроме того, я узнал о связи задачи нахождения порядка точки с задачей дискретного логарифмирования, что важно для понимания основ безопасности криптографических систем на основе эллиптических кривых.

# **Список используемой литературы**

<https://datatracker.ietf.org/doc/html/rfc5639#section-3.3>

<https://habr.com/ru/articles/335906/>