**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 № 4   
по курсу «Криптография»

Группа: М8О-312Б-22

Студент(ка): Л. Д, Андрюшин

Преподаватель: А. В. Борисов

Оценка:

Дата: 25.05.2025

Москва, 2025

**ОГЛАВЛЕНИЕ**

[1 Тема 3](#_Toc158983147)

[2 Задание 3](#_Toc158983148)

[3 Теория 3](#_Toc158983149)

[4 Ход лабораторной работы 6](#_Toc158983150)

[5 Выводы 10](#_Toc158983151)

# **Тема**

Лабораторная работа №4

# **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**

Процессор - Процессор AMD Ryzen 5 5600H with Radeon Graphics, 3301 МГц, ядер: 6, логических процессоров: 12

Оперативная память – 16 ГБ DDR 4

## **Эллиптические кривые**

Эллиптические кривые представляют собой специальный вид кривых, которые описываются уравнением Вейерштрасса:

В криптографии также накладывается дополнительное ограничение на эллиптические кривые: . Кривые, удовлетворяющие этому условию, называются **гладкими**.

## **Операция сложения**

Мы можем также определить на эллиптической кривой алгебру.

**Сложение трех точек** на эллиптической кривой обычно определяется следующим образом: если мы имеем три точки , и на кривой, и они лежат на одной прямой, то их сумма равна нулю (то есть ). Если мы хотим сложить только две точки, скажем и , мы просто берем точку , которая является третьей точкой на прямой через и , и определяем как обратную к . Если же , то точка является точкой пересечения касательной в точке с кривой.

**Обратной точкой** точке будем называть точку , которая является зеркальной версией точки относительно оси .

**Нулевая точка** на эллиптической кривой представляет собой "точку в бесконечности". Она играет роль нейтрального элемента в группе, определенной на кривой. Это означает, что для любой точки на кривой выполняется равенство . Кроме того, если точка находится на кривой, то .

## **Кривые над конечным полем**

В криптографии используется специальный вид кривых, которые определены на конечном поле. Это поле представляет собой кольцо вычетов по модулю простого числа. Такое поле обычно обозначается как и содержит целые положительные числа в диапазоне . Точки на такой кривой должны удовлетворять следующему равенству:

То есть, все операции проводятся в контексте модуля .

На таком кольце определены операции сложения, вычитания, умножения по модулю. Также, над кольцом определена операция дискретного логарифма, который на самом деле представляет собой аналог операции "деления" над полем .

## **Особенности сложения**

Операция сложения в таком поле имеет свои особенности. Если мы складываем две точки и на эллиптической кривой, результатом будет точка , где:

здесь представляет наклон прямой, проходящей через точки и .

## **Порядок**

**Порядок кривой** соответствует количеству точек на кривой, включая точку в бесконечности. В контексте эллиптической кривой, порядок точки означает такое наименьшее положительное число , при котором , где - это точка в бесконечности, а - это результат сложения точки самой с собой раз.

## **Задача дискретного логарифмирования**

В контексте эллиптических кривых, задача дискретного логарифмирования формулируется следующим образом. Пусть даны точки , на эллиптической кривой, и требуется найти такое целое число , которое будет удовлетворять следующему равенству

где означает результат сложения точки с самой собой раз.

Такая задача является на текущей момент "сложной". То есть, не было придумано алгоритма, который смог бы решить поставленную задачу за приемлемое время. Если бы был найден эффективный алгоритм для решения этой задачи, то большинство современных криптосистем стали бы небезопасными.

Сложность задачи дискретного логарифмирования в контексте эллиптических кривых обусловлена тем, что операции на эллиптических кривых отличаются от операций в обычных группах. В частности, операция сложения на эллиптической кривой не является коммутативной, что затрудняет применение многих известных алгоритмов. Кроме того, эффективные алгоритмы для решения задачи дискретного логарифмирования в других группах, таких как группа целых чисел по модулю $p$, не могут быть применены к эллиптическим кривым, что делает задачу дискретного логарифмирования на них особенно сложной.

# **Ход лабораторной работы**

В ходе выполнения лабораторной работы передо мной стояла задача: написать программу, которая бы находила порядок заданной точки на эллиптической кривой над конечным полем по модулю p. Причём задание содержало дополнительное условие — построить такую кривую и подобрать такую точку, чтобы метод полного перебора порядка работал примерно 10 минут на обычном персональном компьютере, то есть около 600 секунд. Это условие сразу делало задачу нетривиальной, поскольку при случайном выборе параметров кривой и точки порядок обычно либо слишком мал (и определяется за считаные секунды), либо слишком велик (и тогда вычисление занимает часы или даже дни).

На первом этапе был реализован базовый код, выполняющий арифметику точек на эллиптической кривой над полем по модулю p. То есть была реализована функция сложения двух точек, а также функция нахождения порядка точки путём последовательного сложения её самой с собой до тех пор, пока результат не превратится в нейтральный элемент (то есть бесконечно удалённую точку, обозначаемую как "O").

После этого стало очевидно, что одного базового кода недостаточно — случайный выбор кривой и точки не даёт подходящего результата. Тогда было принято решение расширить код: сделать автоматический подбор параметров кривой (коэффициентов A и B) и точки P. При этом модуль p был взят равным 1000000007 — это большое простое число, часто используемое в программировании для работы с модульной арифметикой.

Для генерации корректной эллиптической кривой использовалась проверка на невырожденность: дискриминант 4A³ + 27B² по модулю p должен быть отличен от нуля. Случайные значения A и B перебирались до тех пор, пока условие выполнялось. Далее генерировалась точка: случайно выбиралось значение x, подставлялось в уравнение кривой, и проверялось, является ли полученное значение y² квадратичным вычетом по модулю p. Если да — вычислялось значение y, и точка считалась допустимой.

Следующий этап — подбор нужной кривой. Поскольку вычисление порядка — очень затратная по времени операция, в код была встроена система измерения времени выполнения и проверки, в какой диапазон попадает результат. Если время оказалось меньше 500 секунд, программа считала, что порядок слишком мал, и переходила к новой кривой. Если время превышало 700 секунд, программа считала, что перебор идёт слишком долго, и также переходила к следующей кривой. Таким образом, программа продолжала искать такую кривую и точку, которые дали бы порядок, требующий примерно 10 минут вычислений.

В ходе работы было проведено множество запусков программы, иногда подбор занимал довольно длительное время, поскольку попадание в нужный диапазон — редкое событие. Однако в итоге была найдена подходящая кривая с уравнением: y² = x³ + 453563488x + 314744443 по модулю 1000000007, и точка P с координатами (445741754, 821582529). Для этой точки порядок составил 273114839, а время вычислений — 562.73 секунды, что попадает в допустимый диапазон и полностью удовлетворяет требованиям задачи.

Код программы:

import random

import time

P = 10\*\*9 + 7

class Point:

    def \_\_init\_\_(self, x, y):

        self.x = x

        self.y = y

    def \_\_eq\_\_(self, q):

        if q == "O":

            return False

        return self.x == q.x and self.y == q.y

    def \_\_ne\_\_(self, q):

        return not (self == q)

    def \_\_add\_\_(self, q):

        if q == "O":

            return self

        if self == "O":

            return q

        if self.x == q.x and (self.y + q.y) % P == 0:

            return "O"

        if self == q:

            if self.y == 0:

                return "O"

            inv\_denominator = pow(2 \* self.y, P - 2, P)

            m = (3 \* self.x \* self.x + A) \* inv\_denominator % P

        else:

            inv\_denominator = pow(q.x - self.x, P - 2, P)

            m = (q.y - self.y) \* inv\_denominator % P

        x\_r = (m \* m - self.x - q.x) % P

        y\_r = (m \* (self.x - x\_r) - self.y) % P

        return Point(x\_r, y\_r)

    def \_\_str\_\_(self):

        return f"({self.x}, {self.y})"

def is\_quadratic\_residue(n, p):

    if n == 0:

        return True

    return pow(n, (p - 1) // 2, p) == 1

def mod\_sqrt(n, p):

    if not is\_quadratic\_residue(n, p):

        return None

    return pow(n, (p + 1) // 4, p)

def generate\_curve():

    while True:

        global A, B

        A = random.randint(1, P - 1)

        B = random.randint(1, P - 1)

        if (4 \* pow(A, 3, P) + 27 \* pow(B, 2, P)) % P != 0:

            return A, B

def find\_point\_on\_curve(A, B):

    while True:

        x = random.randint(0, P - 1)

        y\_squared = (pow(x, 3, P) + A \* x + B) % P

        y = mod\_sqrt(y\_squared, P)

        if y is not None:

            return Point(x, y)

def compute\_order(P0):

    Q = P0

    n = 1

    start\_time = time.time()

    report\_interval = 10\_000\_000

    while Q != "O":

        Q = Q + P0

        n += 1

        if n % report\_interval == 0:

            elapsed = time.time() - start\_time

            print(f"Проверено порядок: {n}, время: {elapsed:.2f} сек")

        elapsed = time.time() - start\_time

        if elapsed > 700:

            return None, n, elapsed

    elapsed = time.time() - start\_time

    return n, n, elapsed

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

    attempt = 0

    while True:

        attempt += 1

        print(f"\nПопытка {attempt}")

        A, B = generate\_curve()

        print(f"Проверяем кривую: y^2 = x^3 + {A}x + {B} над F\_{P}")

        P0 = find\_point\_on\_curve(A, B)

        print(f"Начальная точка: {P0}")

        order, checked, duration = compute\_order(P0)

        if order is None:

            print(f"Превышено максимальное время (>{700} сек), перебираем снова.")

            continue

        if duration < 500:

            print(f"Время вычисления {duration:.2f} сек меньше 500, пробуем другую кривую.")

            continue

        if 500 <= duration <= 700:

            print(f"\nПодходящая кривая найдена!")

            print(f"Порядок точки: {order}")

            print(f"Время вычислений: {duration:.2f} сек")

            print(f"Кривая: y^2 = x^3 + {A}x + {B} mod {P}")

            print(f"Точка: {P0}")

            break

Вывод программы:

Пробуем кривую y^2 = x^3 + 453563488x + 314744443 mod 1000000007

Точка P = (445741754, 821582529)

Вычисляем порядок точки P...

***\*тут идет куча проверок, которые я не буду копировать\****

Порядок точки P: 273114839

Время вычислений: 562.73

# **Выводы**

В ходе лабораторной работы была реализована программа на языке Python, позволяющая определить порядок заданной точки на эллиптической кривой над конечным полем. Путём подбора параметров кривой и начальной точки удалось добиться такого порядка точки, при котором полный перебор укладывается в заданное время, которое примерно равнялось 10 минутам. Были исследованы свойства кривых, удовлетворяющих условиям задачи, и произведена оптимизация начальных параметров для ускорения расчётов. Полученные результаты подтвердили корректность реализации алгоритма и позволили на практике понять принцип работы операций на эллиптических кривых.