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

Криптография на эллиптических кривых

2 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.

3 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:

$y2=x3+ax+b$y2=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 $P\neq Q$P
     $\lambda=y2-y1x2-x1$
     $x3=\lambda2-x1-x2$
     $y3=\lambda(x1-x3)-y1$
   - Если $P=Q$ (удвоение точки):
     $\lambda=3x12+a2y1$
     $x3=\lambda2-2x1$
     $y3=\lambda(x1-x3)-y1$

2. 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:

$y2 \equiv x3 + ax + b$ (mod p)

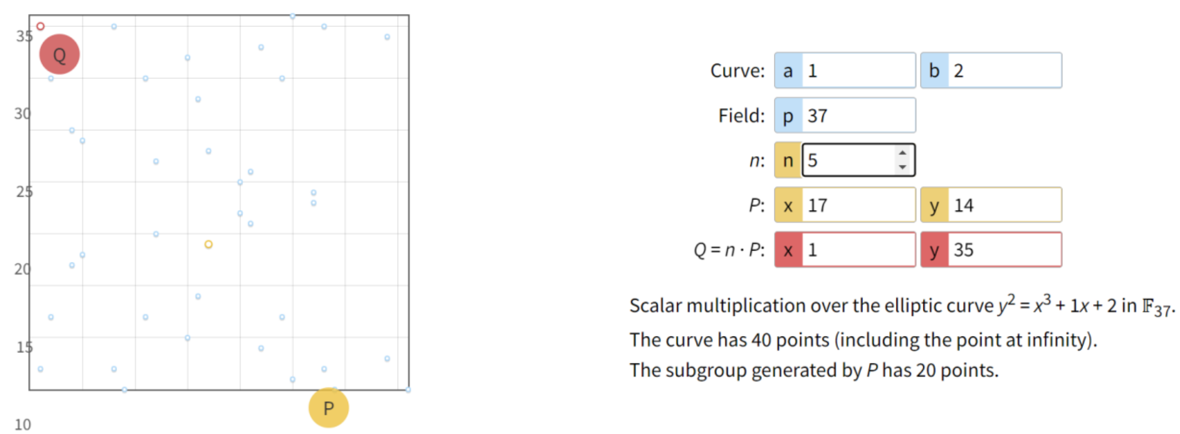
Здесь арифметика ведется по модулю pp. Это означает, что все вычисления с координатами точек происходят в поле вычетов по модулю pp. В таком контексте добавление и умножение точек выполняются с использованием операций по модулю pp.

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.

4 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 = 1098318681686843663097589633089531377087289 52

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 = 1098318681686843663097589633089531377087 28952

    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 $O(\sqrt{n})$. 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

5 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.

6 Список используемой литературы

https://datatracker.ietf.org/doc/html/rfc5639#section-3.3
https://habr.com/ru/articles/335906/