Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 3
по курсу «Криптография»

Студент: Пивницкий Д.С.
Группа: М80-306Б-19
Преподаватель: Борисов А. В.
Оценка:

Москва, 2022

1. Постановка задачи

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.

2. Метод решения

Я использовал каноническую форму эллиптической кривой:
y^2 = x^3 + ax + b.

Коэффициенты A и B были выбраны случайно. Модуль кривой p подбирался вручную, пока время вычисления порядка точки не стал удовлетворять условию. Через пару попыток был подобран p = 19727, такой, что порядок точки находится за 10 минут.

Алгоритм работы: Полным перебором нахожу все точки, принадлежащие кривой. Случайным образом выбираю одну из точек, затем складываю её, пока сумма не станет равна (0, 0). Перебор работает за квадрат. Далее будет приведён способ, как ускорить решение задачи.

3. Полученные результаты

```
(py37) →  crypto python lab3.py
Order curve = 19520
Order point (8585, 9591): 2404
Time: 600.6791009902954 sec.
(py37) →  crypto
```

4. Код программы

```
import time
import random

A = 18603487494924907898232888139306253817 60
B = 20016375066713848331718186731490628059 74

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

def extended_euclidean_algorithm(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_of(n, p):
    gcd, x, y = extended_euclidean_algorithm(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 add_points(p1, p2, p):
    if p1 == (0, 0):
        return p2
    elif p2 == (0, 0):
        return p1
    elif p1[0] == p2[0] and p1[1] != p2[1]:
        return (0, 0)

    if p1 == p2:
        s = ((3 * p1[0] ** 2 + (A % p)) * inverse_of(2 * p1[1], p)) % p
    else:
        s = ((p1[1] - p2[1]) * inverse_of(p1[0] - p2[0], p)) % p
    x = (s ** 2 - 2 * p1[0]) % p
    y = (p1[1] + s * (x - p1[0])) % p
    return (x, -y % p)

def order_point(point, p):
    i = 1
    check = add_points(point, point, p)
    while check != (0, 0):
        check = add_points(check, point, p)
        i += 1
    return i

def main():
    p = 19727

    points = []
    start = time.time()
    for x in range(p):
```

```
        for y in range(p):
            if elliptic_curve(x, y, p):
                points.append((x, y))

    point = random.choice(points)
    print(f"Order curve = {len(points)}")
    print(f"Order point {point}: {order_point(point, p)}")
    print("Time: {} sec.".format(time.time() - start))

if __name__ == "__main__":
    main()
```

5. Выводы

Выполнив данную лабораторную работу, я разобрался с эллиптическими кривыми, алгоритмами, связанными с ними и то, как они применяются в криптографии. Решил данную задачу методом полного перебора, но есть и другие способы. Для ускорения решения задачи полного перебора используются, например, алгоритм Шуфа с теоремой Хассе. Сложность O(log(q)^8), где q – число элементов поля. Еще существует метод комплексного умножения, который позволяет более эффективно находить кривые с заданным количеством точек. Однако в отличие от алгоритма Шуфа, метод комплексного умножения работает только при выполнении определенных условий.