Московский авиационный институт   
(государственный технический университет)   
  
Факультет прикладной математики   
  
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Криптография»:

8 факультет, 3 курс

Студент: Стрыгин Д.Д.

Группа: М8О-306Б-19, №25

Преподаватель: Борисов А.В.

**Задача:**

1.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.

**Теоретическое введение:**

**Описание работы:**

Для решения поставленной задачи я использовал язык программирования python. В программе я написал проверку на корректность данных в соответствии с условиями:

1.Коэффициент p – простое число

2.Коэффициенты a и b удовлетворяют неравенству



Затем, в программе определяется массив из точек эллиптической кривой, после чего берётся любая точка, принадлежащая этой кривой, и для неё высчитывается порядок. С помощью библиотеки python для работы со временем выполнения программы, я фиксировал время, затем путём проб и подбора по дихотомии смог получить время, приближенной к 10 минутам.

**Листинг программы:**

import sys

import random

import math

import time

import sympy

def elipit\_func(x, y, p, a, b):

    return (y \*\* 2) % p == ((x \*\* 3) + (a % p) \* x + (b % p)) % p

def find\_points(p, a, b):

    points = []

    for x in range(p):

        for y in range(p):

            if elipit\_func(x, y, p, a, b):

                points.append((x, y))

    return points

def ABtest(a, b):

    if 4 \* a \*\* 3 + 27 \* b \*\* 2 == 0:

        print("Fatal error", file=sys.stderr)

        quit()

def test\_p(p):

    if not(sympy.isprime(p)):

        print("Fatal error", file=sys.stderr)

        quit()

def list\_to\_float(lst):

    for i in range(len(lst)):

        lst[i] = float(lst[i])

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(

            '{} не имеет мультпликативную инверсию'

            'Mod {}'.format(n, p))

    else:

        return x % p

def add\_points(p1, p2, p, a):

    x1, y1 = p1[0], p1[1]

    x2, y2 = p2[0], p2[1]

    if p1 == (0, 0):

        return p2

    elif p2 == (0, 0):

        return p1

    elif x1 == x2 and y1 != y2:

        return (0, 0)

    if p1 == p2:

        m = ((3 \* x1 \*\* 2 + (a % p)) \* inverse\_of(2 \* y1, p)) % p

    else:

        m = ((y1 - y2) \* inverse\_of(x1 - x2, p)) % p

    x3 = (m \*\* 2 - x1 - x2) % p

    y3 = (y1 + m \* (x3 - x1)) % p

    return [x3, -y3 % p]

def point\_order(point, p, a):

    i = 1

    new\_point = add\_points(point, point, p, a)

    while new\_point != (0, 0):

        new\_point = add\_points(new\_point, point, p, a)

        i += 1

    return i

def main():

    elipit = input("Введите значения коэффициентов a и b: ").split()

    p = int(input("Введите коэффициент p эллиптической кривой: "))

    test\_p(p)

    list\_to\_float(elipit)

    a, b = elipit

    ABtest(a, b)

    start = time.time()

    points = find\_points(p, a, b)

    point = random.choice(points)

    order = point\_order(point, p, a)

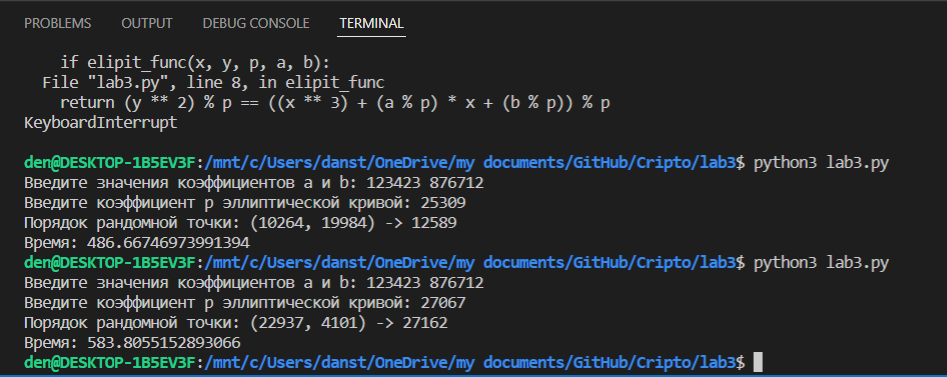
    print("Порядок рандомной точки: {0} -> {1}".format(point, order))

    print("Время: {0}".format(time.time() - start))

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

    main()

**Результаты тестирования:**



**Трудности, встреченные в процессе:**

Проблема возникла на этапе подбора значений, в итоге принял решение построить график для увеличения числа коэффициента p, по которому рассчитывался примерный диапазон значений, необходимых для достижения 10 минут.

**Выводы:**

Эллиптические кривые обеспечивают безопасное шифрование на основе открытых ключей.

**Литература:**

* Электронный ресурс «Доступно о криптографии на эллиптических кривых» URL: <https://habr.com/ru/post/335906/>