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 №3
по курсу «Криптография»

Студент:	Хренникова А. С.
Группа:	М8О-308Б-19
Преподаватель:	Борисов. А. В.
Подпись:	
Оценка:	
Дата:	

Москва, 2022

Лабораторная работа №3

Задача:

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.

1 Описание:

Каноническая форма эллиптической кривой: $y^2 = x^3 + ax + b$, при этом $4a^3 + 27b^2$ не принимает 0 значение – обычная формулировка Вейерштрасса.

Эллиптическая кривая, определенная над конечным полем, имеет конечное количество точек. Порядок группы – количество точек в группе. Полный перебор для всех возможных x выполняется достаточно долго, особенно если P – довольно большое число. Для ускорения существует алгоритм Шуфа.

2 Исходный код:

```
import time
import random
import math
import numpy as np
import matplotlib.pyplot as plt

A = random.randint(1000000000, 10000000000)
B = random.randint(1000000000, 10000000000)
P = 103

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

def print_elliptic_curve():
  print("y^2 = x^3 + {0} * x + {1} (mod {2})".format(A % P, 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):
  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):
  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
  else:
    s = ((p1[1] - p2[1]) * inverse_of(p1[0] - p2[0])) % P

  x = (s ** 2 - 2 * p1[0]) % P
  y = (p1[1] + s * (x - p1[0])) % P
  return (x, -y % P)

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

def step():
  print_elliptic_curve()
  points = []
  start_time = time.time()
  for x in range(0, P):
    for y in range(0, P):
      if elliptic_curve(x, y):
        points.append((x, y))
  print("Порядок кривой: {0}".format(len(points)))
  point = random.choice(points)
  print("Порядок точки P({0}, {1}): {2}".format(point[0], point[1], order_poi
nt(point)))
  time_value = time.time() - start_time
  print("Потраченное время: {} сек.".format(time_value))
```

```
  return time_value

def is_simple_number(number):
  is_find = True
  for i in range(2 ,int(math.sqrt(number))+1):
    if(number % i == 0):
      is_find = False
      break
  return is_find

def gen_next_simple_number(start_point):
  while(not(is_simple_number(start_point))):
    start_point += 1
  return start_point

if __name__ == '__main__':
  print("----------------------------------------------------")
  time_value = 0
  iteration = 1
  while (time_value < 600):
    P = gen_next_simple_number(P + iteration * 1000)
    time_value = step()
    iteration += 1
    print("----------------------------------------------------")
```

```
-----------------------------------------------------
y^2 = x^3 + 164 * x + 726 (mod 1103)
Порядок кривой: 1147
Порядок точки P(432, 883): 146
Потраченное время: 1.2440531253814697 сек.
-----------------------------------------------------
y^2 = x^3 + 1441 * x + 1405 (mod 3109)
Порядок кривой: 3131
Порядок точки P(2556, 2053): 4913
Потраченное время: 10.514172792434692 сек.
-----------------------------------------------------
y^2 = x^3 + 4391 * x + 4971 (mod 6113)
Порядок кривой: 6119
Порядок точки P(1967, 2598): 1931
Потраченное время: 40.246984004974365 сек.
-----------------------------------------------------
y^2 = x^3 + 6125 * x + 3568 (mod 10133)
Порядок кривой: 10232
Порядок точки P(9946, 5787): 499
Потраченное время: 111.66374516487122 сек.
-----------------------------------------------------
y^2 = x^3 + 9951 * x + 10644 (mod 15137)
Порядок кривой: 15124
Порядок точки P(10946, 9192): 7430
Потраченное время: 250.19150471687317 сек.
-----------------------------------------------------
y^2 = x^3 + 14031 * x + 10164 (mod 21139)
Порядок кривой: 21369
Порядок точки P(11636, 7287): 28701
Потраченное время: 480.90807604789734 сек.
-----------------------------------------------------
y^2 = x^3 + 6473 * x + 9469 (mod 28151)
Порядок кривой: 27890
```

```
Порядок точки P(22802, 4638): 19200
Потраченное время: 859.724196434021 сек.
-------------------------------------------------------
```

Примерно за 10 минут получается такой результат:

```
y^2 = x^3 + 7413 * x + 16162 (mod 23993)
Порядок кривой = 23927
Порядок точки З(11901, 10065): 2990
Потраченное время: 621.3101108074188
```

3 Выводы:

Для С помощью эллиптических кривых можно построить асимметрическую криптосистему, где закрытым ключом является число d(выбранное из множества от 1 до n-1, где n – порядок подгруппы), а открытым ключом является точка H=dG(где G - базовая точка подгруппы). При этом, даже если известны H и G, то поиск закрытого ключа d является «сложной» задачей, потому что требует решения задачи дискретного логарифмирования.

Москва, 2022