Лабораторная работа №4 по курсу криптографии

Выполнил студент группы М8О-308Б-18 Коростелев Дмитрий

Условие

Подобрать такую эллиптическую кривую над конечным простым полем порядка p, такую, порядок точки которой полным перебором находится за 10 минут на ПК. Упомянуть в отчёте какие алгоритмы и теоремы существуют для облегчения и ускорения решения задачи полного перебора

Метод решения

Данную задачу будем решать методом полного перебора. Поиск будем продолжать до того момента, пока не найдем нужную эллиптическую кривую, меняя значения коэффициента P на каждой итерации. P – коэффициент на который делится эллиптическая кривая и выражено простым числом. После каждой неудачной итерации подбирается следующее простое число, которое усложняет поиск порядка точки эллиптической кривой.

Коэффициенты A и B напечатаны абсолютно случайным образом.

Точки, принадлежащие эллиптической кривой, будем искать полным перебором. После нахождения точек, выбираем из них случайную и ищем ее порядок, сохраняя временные метки. Порядок точки находится сложением точки с самой собой до того момента, пока значение не станет равным нулю. Алгоритм сложения точек на эллиптической кривой взят из открытого источника.

```
import time
import random
import math


A = 1860348357352452346236348964875285235
B = 2001637236353457375457923762396729387
P = 103

Ap = A % P
Bp = B % P

def elliptic_curve(x, y):
    global Ap, Bp
    return (y ** 2) % P == (x ** 3 + (Ap) * x + (Bp)) % P


def print_elliptic_curve():
    global Ap, Bp
    print("y^2 = x^3 + {0} * x + {1} (mod {2})".format(Ap, Bp, 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):
    global Ap
    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 + (Ap)) * 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 {0}: {1}".format(point, order_point(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 * 3000)
        Ap = A % P
        Bp = B % P
        time_value = step()
        iteration += 1
        print("----------------------------------------------------")
```

Результат работы программы

```
D:\ProgramFiles\Anaconda\python.exe elliptic_curve.py
-----------------------------------------------------
y^2 = x^3 + 729 * x + 669 (mod 3109)
Порядок кривой: 3215
Порядок точки (1918, 759): 1760
Потраченное время: 9.85568881034851 сек.
-----------------------------------------------------
y^2 = x^3 + 403 * x + 3581 (mod 9109)
Порядок кривой: 9102
Порядок точки (52, 1755): 4353
Потраченное время: 87.71361327171326 сек.
-----------------------------------------------------
y^2 = x^3 + 4655 * x + 334 (mod 18119)
Порядок кривой: 17939
Порядок точки (8132, 9009): 57559
Потраченное время: 310.0321047306061 сек.
-----------------------------------------------------
y^2 = x^3 + 4376 * x + 5136 (mod 30119)
Порядок кривой: 29927
Порядок точки (20176, 386): 967
Потраченное время: 899.2285053730011 сек.
-----------------------------------------------------
```

После 4ых итераций удалось найти эллиптическую кривую, удовлетворяющую условия задания, а поиск порядка точки данной прямой занимает примерно 15 минут, при относительно небольшом числе P = 30119

Выводы

Для снижения вычислительной сложности решения задачи полного перебора часто применяется теорема Хассе, которая используется в алгоритмах больших и маленьких шагов, Шуфа и Шуфа — Элкиса — Аткина. Данные алгоритмы позволяют значительно снизить сложность перебора, вплоть до $O(log^4q)$, где q – число элементов поля.