Отчёт по лабораторной работе №7

Дискретное логарифмирование в конечном поле

Хитяев Евгений Анатольевич НПМмд-02-21

Содержание

List of Figures

1 Цель работы

Изучение задачи дискретного логарифмирования.

2 Теоретические сведения

Пусть в некоторой конечной мультипликативной абелевой группе G задано уравнение

$$g^x = a$$

Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа x, удовлетворяющего уравнению. Если оно разрешимо, у него должно быть хотя бы одно натуральное решение, не превышающее порядок группы. Это сразу даёт грубую оценку сложности алгоритма поиска решений сверху — алгоритм полного перебора нашёл бы решение за число шагов не выше порядка данной группы.

Чаще всего рассматривается случай, когда группа является циклической, порождённой элементом g. В этом случае уравнение всегда имеет решение. В случае же произвольной группы вопрос о разрешимости задачи дискретного логарифмирования, то есть вопрос о существовании решений уравнения , требует отдельного рассмотрения.

2.1 р-алгоритм Полларда

- Вход. Простое число p, число a порядка r по модулю p, целое число bб $1 < b < p$; отображение f, обладающее сжимающими свойствами и сохраняющее вычислимость логарифма.

- Выход. показатель x, для которого $a^x = b (mod p)$, если такой показатель существует.

1. Выбрать произвольные целые числа u, v и положить $c = a^u b^v (mod p), d = c$
2. Выполнять $c=f(c)(mod p), d=f(f(d))(mod p), вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения равенства $c = d (mod p)$
3. Приняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат x или “Решения нет”.

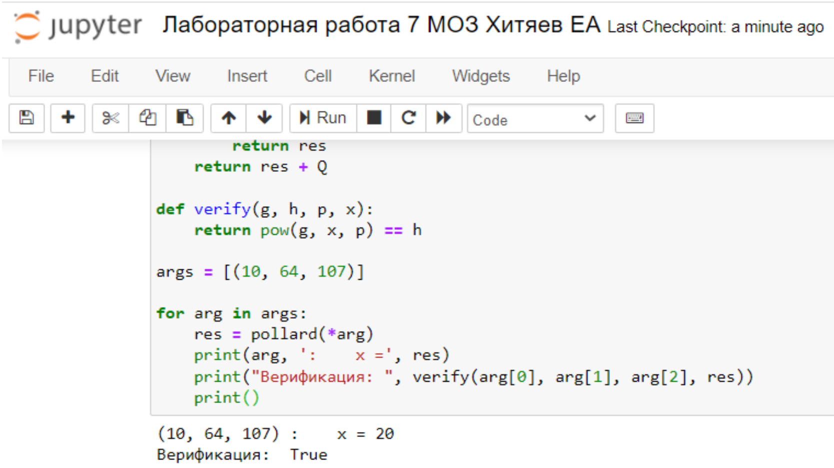
3 Выполнение работы

3.1 Реализация алгоритмов на языке Python

```
from math import gcd

ag = 1
bg = 1

def f(x, n):
    return (x*x+5)%n

def method(n, a, b, d):
    a = f(a, n)%n
    b = f(f(b,n), n)%n
    d = gcd(a-b, n)
    if 1 < d < n:
        p = d
        print(p)
        exit()
    if d == n:
        print("Делитель не найден")
    if d == 1:
        global ag
```

```
        ag = b
        method(n, a, b, d)

def main():
    n = 1359331
    c = 1
    a = c
    b = c
    a = f(a, n)%n
    b = f(a, n)%n
    d = gcd(a-b, n)
    if 1 < d < n:
        p = d
        print(p)
        exit()
    if d == n:
        pass
    if d == 1:
        method(n, a, b, d)

main()
```

3.2 Контрольный пример




```
        return res
    return res + Q

def verify(g, h, p, x):
    return pow(g, x, p) == h

args = [(10, 64, 107)]

for arg in args:
    res = pollard(*arg)
    print(arg, ':    x =', res)
    print("Верификация: ", verify(arg[0], arg[1], arg[2], res))
    print()
```

```
(10, 64, 107) :    x = 20
Верификация:  True
```

Figure 3.1: Пример работы алгоритма

Таким образом, число 1181 является нетривиальным делителем числа 1359331.

4 Выводы

В ходе выполнения работы мне удалось изучить задачу разложения на множители и p-алгоритм Полларда, а также реализовать данный алгоритм программно на языке Python.

Список литературы

1. Алгоритмы тестирования на простоту и факторизации
2. P-метод Полларда