Математические основы защиты информации и информационной безопасности. Лабораторная работа №7

Дискретное логарифмирование в конечном поле

Содержание

# 1 Цель работы

Изучение алгоритма p-Полларда для задач дискретного логарифмирования.

# 2 Задачи

Реализовать программно алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования

# 3 Теоретические сведения

Задача обращения функции в некоторой конечной мультипликативной группе .

Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем. Эффективные алгоритмы для решения задачи дискретного логарифмирования в общем случае неизвестны.

Для заданных и решение уравнения называется дискретным логарифмом элемента по основанию .

## 3.1 p-метод Полларда для задач дискретного логарифмирования

* Вход. Простое число , число порядка по модулю , целое число , ; отображение , обладающее сжимающими свойствами и сохраняющее вычислимость логарифма.
* Выход. Показатель , для которого , если такой показатель существует.

1. Выбрать произвольные числа и положить
2. Выполнять , вычисляя при этом логарифмы для и как линейные функции от по модулю , до получения равенства
3. Приравняв логарифы для и , вычислить логарифм решением сравнения по модулю . Результат: или “Решений нет”.

Подробнее об алгоритме: [2]

# 4 Выполнение работы

## 4.1 Реализация алгоритмов

def ext\_euclid(a, b):  
 if b == 0:  
 return a, 1, 0  
 else:  
 d, xx, yy = ext\_euclid(b, a % b)  
 x = yy  
 y = xx - (a // b) \* yy  
 return d, x, y  
  
  
def inverse(a, n):  
 return ext\_euclid(a, n)[1]  
  
  
def xab(x, a, b, value):  
 (G, H, P, Q) = value  
 sub = x % 3  
  
 if sub == 0:  
 x = x \* value[0] % value[2]  
 a = (a + 1) % Q  
  
 if sub == 1:  
 x = x \* value[1] % value[2]  
 b = (b + 1) % value[2]  
  
 if sub == 2:  
 x = x \* x % value[2]  
 a = a \* 2 % value[3]  
 b = b \* 2 % value[3]  
  
 return x, a, b  
  
  
def verify(g, h, p, x):  
 return pow(g, x, p) == h  
  
  
def pollard(G, H, P):  
 Q = int((P - 1) // 2)  
  
 x = G \* H  
 a = 1  
 b = 1  
  
 X = x  
 A = a  
 B = b  
  
 for i in range(1, P):  
 x, a, b = xab(x, a, b, (G, H, P, Q))  
  
 X, A, B = xab(X, A, B, (G, H, P, Q))  
 X, A, B = xab(X, A, B, (G, H, P, Q))  
  
 if x == X:  
 break  
  
 nom = a - A  
 denom = B - b  
  
 res = (inverse(denom, Q) \* nom) % Q  
  
 if verify(G, H, P, res):  
 return res  
  
 return res + Q  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 args = [(10, 64, 107)]  
 for arg in args:  
 res = pollard(\*arg)  
 print("{} \*\* {} ≡ {} (mod {})".format(arg[0], res, arg[1], arg[2]))  
 print("Verify result: ", end="")  
 if verify(arg[0], arg[1], arg[2], res):  
 print("verified")  
 else:  
 print("not verified")

## 4.2 Пример алгоритма p-Полларда для задач дискретного логарифмирования

На рис. 1 представлены результаты работы p-метода Полларда для задач дискретного логарифмирования:

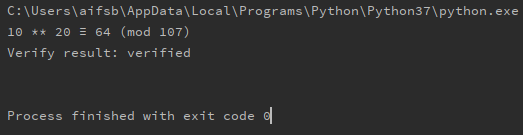


Figure 1: Пример алгоритма p-Полларда

# 5 Выводы

В ходе выполнения работы был успешно изучен p-метод Полларда для задач дискретного логарифмирования, а также был реализован программно программно на языке Python.

# Список литературы

1. Ро-алгоритм Полларда [Электронный ресурс]. Википедия, 2021. URL: <https://ru.wikipedia.org/wiki/Ро-алгоритм_Полларда>.

2. Дискретное логарифмирование [Электронный ресурс]. Википедия, 2021. URL: <https://ru.wikipedia.org/wiki/Дискретное_логарифмирование>.