Отчёт по лабораторной работе №7

Дискретное логарифмирование в конечном поле

Кодже Лемонго Арман

Содержание

Список иллюстраций

# Цель работы

Целью данной является изучение задачи дискретного логарифмирования.

# Теоретические сведения

Дискретное логарифмирование (DLOG) — задача обращения функции g^{x} в некоторой конечной мультипликативной группе G .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 g и a решение x уравнения g^{x}=a называется дискретным логарифмом элемента a по основанию g. В случае, когда G является мультипликативной группой кольца вычетов по модулю m, решение называют также индексом числа a по основанию g. Индекс числа a по основанию g гарантированно существует, если g является первообразным корнем по модулю m.

## p-алгоритм Поллрада

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

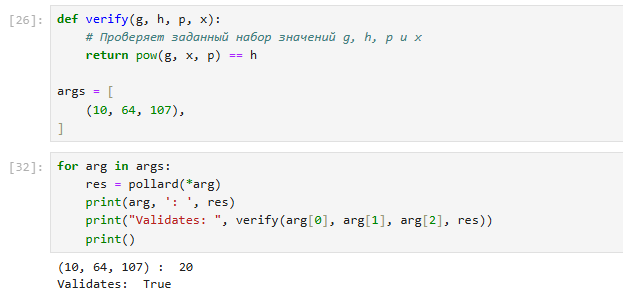
1. Выбрать произвольные целые числа и положить
2. Выполнять $c=f(c)(mod p), d=f(f(d))(mod p), вычисляя при этом логарифмы для и как линейные функции от по модулю , до получения равенства
3. Приняв логарифмы для и , вычислить логарифм решением сравнения по модулю . Результат или РЕШЕНИЯ НЕТ.

# Выполнение работы

## Реализация алгоритма на языке Python

def ext\_euclid(a, b):  
 # Extended Euclidean Algorithm  
 # param a, param 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):  
 #Inverse of a in mod n  
 #param a, param n:  
 return ext\_euclid(a, n)[1]  
  
def xab(x, a, b, xxx\_todo\_changeme):  
 # Pollard Step  
 # param x, param a, param b:  
   
 (G, H, P, Q) = xxx\_todo\_changeme  
 sub = x % 3 # Subsets  
  
 if sub == 0:  
 x = x\*xxx\_todo\_changeme[0] % xxx\_todo\_changeme[2]  
 a = (a+1) % Q  
  
 if sub == 1:  
 x = x \* xxx\_todo\_changeme[1] % xxx\_todo\_changeme[2]  
 b = (b + 1) % xxx\_todo\_changeme[2]  
  
 if sub == 2:  
 x = x\*x % xxx\_todo\_changeme[2]  
 a = a\*2 % xxx\_todo\_changeme[3]  
 b = b\*2 % xxx\_todo\_changeme[3]  
  
 return x, a, b  
  
def pollard(G, H, P):  
  
 Q = int((P - 1) // 2) # подгруппа  
  
 x = G\*H  
 a = 1  
 b = 1  
  
 X = x  
 A = a  
 B = b  
  
 # Не используйте здесь range(). Это делает алгоритм удивительно медленным.  
 for i in range(1, P):  
 # Кому нужна сквозная ссылка, когда у вас есть Python!!! ;  
 # Hedgehog  
 x, a, b = xab(x, a, b, (G, H, P, Q))  
 # Rabbit  
 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  
  
 # print nom, denom  
 # Для правильного вычисления дроби необходимо вычислить обратное значение mod q  
 res = (inverse(denom, Q) \* nom) % Q  
  
 # так никто не делает но все же...  
 if verify(G, H, P, res):  
 return res  
  
 return res + Q  
  
def verify(g, h, p, x):  
 # Проверяет заданный набор значений g, h, p и x  
 return pow(g, x, p) == h  
  
args = [ (10, 64, 107),]  
  
for arg in args:  
 res = pollard(\*arg)  
 print(arg, ': ', res)  
 print("Validates: ", verify(arg[0], arg[1], arg[2], res))  
 print()

## Контрольный пример



Работа алгоритма

# Выводы

в конце нашего лабораторная работа, я изучил задачу дискретного логарифмирования.

# Список литературы

1. [Дискретный логарифм](https://translated.turbopages.org/proxy_u/en-ru.ru.f8961c64-6751851f-1225bc62-74722d776562/https/en.wikipedia.org/wiki/Discrete_logarithm)
2. [Вычислительная сложность и приложения в криптографии](https://ru.ruwiki.ru/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
3. [Дискретное логарифмирование в конечномерной алгебре над полем](https://cyberleninka.ru/article/n/diskretnoe-logarifmirovanie-v-konechnomernoy-algebre-nad-polem)