Отчет по лабораторной работе №7

Дискретное логарифмирование в конечном поле

Бурдина Ксения Павловна

9 декабря 2023

Содержание

# 1 Цель работы

Целью данной работы является освоение дискретного логарифмирования в конечном поле, которое применяется во многих алгоритмах криптографии с открытым ключом.

# 2 Задание

1. Изучить алгоритм дискретного логарифмирования в конечном поле.
2. Реализовать представленный алгоритм и вычислить логарифм по заданным числам .

# 3 Теоретическое введение

Задача дискретного логарифмирования, как и задача разложения на множители, применяется во многих алгоритмах криптографии с открытым ключом. Предложенная в 1976 году У. Диффи и М. Хеллманом для установления сеансового ключа, эта задача послужила основой для создания протоколов шифрования и цифровой подписи, доказательств с нулевым разглашением и других криптографических протоколов.

Пусть над некоторым множеством произвольной природы определены операции сложения “+” и умножения “”. Множество называется *кольцом*, если вполняются следующие условия:

* Сложение коммутативно: для любых ;
* Сложение ассоциативно: для любых ;
* Существует нулевой элемент такой, что для любого ;
* Для каждого элемента существует противоположный элемент , такой, что ;
* Умножение дистрибутивно относительно сложения:
* для любых .

Если в кольце умножение коммутативно: для любых , то кольцо называется *коммутативным*.

Если в кольце умножение ассоциативно: для любых , то кольцо называется *ассоциативным*.

Если в кольце существует едининым элемент такой, что для любого , то кольцо называется кольцом с единицей.

Если в ассоциативном, коммутативном кольце с единицей для каждого ненулевого элемента существует обратный элемент такой, что , то кольцо называется *полем*.

Пусть . Целые числа и называются *сравнимыми по модулю m* (обозначается ( )), если разность делится на . Некоторые свойства отношения сравнимости:

* *Рефлексивность*: ( ).
* *Симметричность*: если ( ), то ( ).
* *Транзитивность*: если ( ) и ( ), то ( ).

Отношение, обладающее свойством рефлесивности, симметриности и транзитивности, называется *отношением эквивалентности*. Отношение сравнимости является отношением эквивалентности на множестве целых чисел [[2]](https://esystem.rudn.ru/pluginfile.php/2089897/mod_folder/content/0/mathsec_lection12-message-integrity-authentication.pdf?forcedownload=1).

Отношение эквивалентности разбивает множество, на котором оно определено, на *классы эквивалентности*. Любые два класса эквивалентности либо не пересекаются, либо совпадают.

Классы эквивалентности, определяемые отношением сравнимости, называются *классами вычетов по модулю m*. Класс вычетов, содержащий число , обознаается ( ) или и представляет собой множество чисел вида , где ; число называется представителем этого класса вычетов.

Множество классов вычетов по модулю обозначается , состоит ровно из элементов и относительно операций сложения и умножения является *кольцом классов вычетов* по модулю .

**Пример**. Если , то , где - множество всех четных чисел, - множество всех нечетных чисел.

Обозначим , - простое целое число и назовем конечным полем из элементов. Задача дискретного логарифмирования в конечном поле формулируется так: для данных целых чисел и , , , найти логарифм - такое целое число , что ( ) (если такое число существует). По аналогии с вещественными числами используется обозначение .

Безопастность соответствующих криптосистем основана на том, что, зная числа вычислить ( ) легко, а решить задачу дискретного логарифмирования трудно. Рассмотрим р-Метод Полларда, который можно применить и для задач дискретного логарифмирования. При этом случайное отображение должно обладать не только сжимающими свойствами, но и вычислимостью логарифма (логарифм числа можно выразить через неизвестный логарифм и ). Для дискретного логарифмирования в качестве случайного отображения чаще всего используются ветвящиеся отображения, например:

При имеем , при имеем .

## 3.1 Алгоритм, реализующий р-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.

* выбрать произвольные целые числа и положить ( ),
* выполнять , , вычисляя при этом логарифмы для и как линейные функции от по модулю , до получения равенства ( )
* приравняв логарифмы для и , вычислить логарифм решением сравнения по модулю . Результат: или “Решений нет”.

**Пример** [[1]](https://intuit.ru/studies/courses/552/408/lecture/9350). Решим задачу дискретного логарифмирования ( ), используя р-Метод Полларда. Порядок числа по модулю равен .

Выберем отображение ( ) при , ( ) при . Пусть . Результаты вычислений запишем в таблицу:

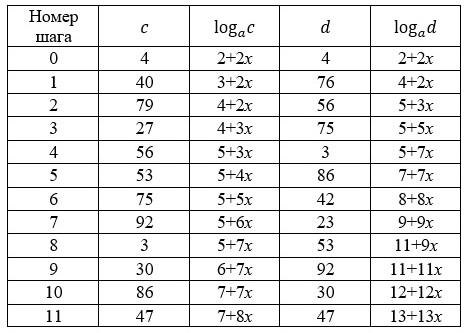


Схема работы алгоритма

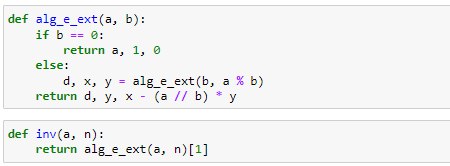
Приравниваем логарифмы, полученные на 11-м шаге: ( ). Решая сравнение первой степени, получаем: ( ).

Проверка: ( ).

# 4 Ход выполнения лабораторной работы

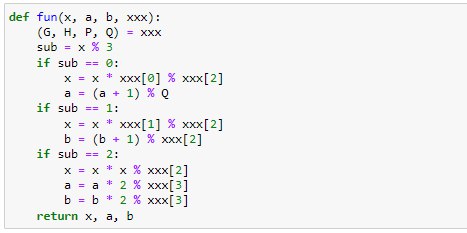
Для реализации рассмотренного алгоритма разложения чисел на множители будем использовать среду JupyterLab. Выполним необходимую задачу.

1. Пропишем алгоритм Евклида, который был показан в предыдущих лабораторных работах, а также запишем функцию для вывода его инверсивного значения:



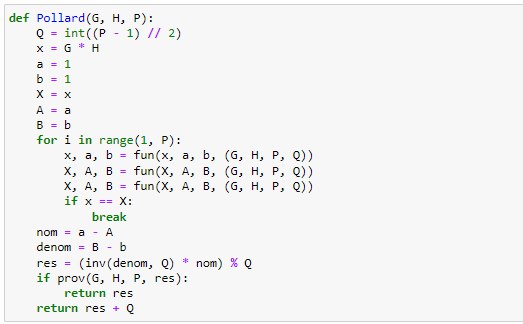
Расширенный алгоритм Евклида

1. Также пропишем функцию для подсчета значений при выполнении алгоритма Полларда:



Вспомогательная функция

1. Запишем алгоритм, реализующий *р-метод Полларда*, с помощью следующей функции:



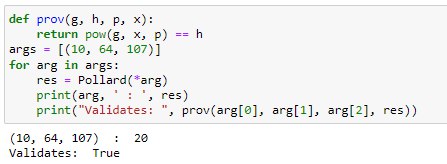
р-Метод Полларда

Здесь на вход подается простое число , число порядка по модулю , целое число ; отображение , обладающее сжимающими свойствами и сохраняющее вычислимость логарифма. Необходимо выполнить следующее:

* выбрать произвольные целые числа и положить ( ),
* выполнять , , вычисляя при этом логарифмы для и как линейные функции от по модулю , до получения равенства ( )
* приравняв логарифмы для и , вычислить логарифм решением сравнения по модулю . Результат: или “Решений нет”.

По итогу при вызове функции мы получим показатель , для которого ( ), если такой показатель существует.

1. Проверим корректность работы алгоритма для заданных сведений. Для этого запишем условие примера с помощью следующей функции:



Пример работы алгоритма

При вызове данной функции видим, что получаем то же число, что было описано в примере. То есть ( ) для задачи дискретного логарифмирования ( ).

# 5 Листинг программы

def alg\_e\_ext(a, b):  
 if b == 0:  
 return a, 1, 0  
 else:  
 d, x, y = alg\_e\_ext(b, a % b)  
 return d, y, x - (a // b) \* y  
  
def inv(a, n):  
 return alg\_e\_ext(a, n)[1]  
  
def fun(x, a, b, xxx):  
 (G, H, P, Q) = xxx  
 sub = x % 3  
 if sub == 0:  
 x = x \* xxx[0] % xxx[2]  
 a = (a + 1) % Q  
 if sub == 1:  
 x = x \* xxx[1] % xxx[2]  
 b = (b + 1) % xxx[2]  
 if sub == 2:  
 x = x \* x % xxx[2]  
 a = a \* 2 % xxx[3]  
 b = b \* 2 % xxx[3]  
 return x, a, b  
  
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 = fun(x, a, b, (G, H, P, Q))  
 X, A, B = fun(X, A, B, (G, H, P, Q))  
 X, A, B = fun(X, A, B, (G, H, P, Q))  
 if x == X:  
 break  
 nom = a - A  
 denom = B - b  
 res = (inv(denom, Q) \* nom) % Q  
 if prov(G, H, P, res):  
 return res  
 return res + Q  
  
def prov(g, h, p, x):  
 return pow(g, x, p) == h  
args = [(10, 64, 107)]  
for arg in args:  
 res = Pollard(\*arg)  
 print(arg, ' : ', res)  
 print("Validates: ", prov(arg[0], arg[1], arg[2], res))

# 6 Выводы

В ходе работы мы изучили и реализовали дискретное логарифмирование в конечном поле.

# 7 Список литературы

1. Фороузан Б. А. Криптография и безопасность сетей. - М.: Интернет-Университет Информационных Технологий : БИНОМ. Лаборатория знаний, 2010. - 784 с. [[1]](https://intuit.ru/studies/courses/552/408/lecture/9350)
2. Методические материалы курса [[2]](https://esystem.rudn.ru/pluginfile.php/2089897/mod_folder/content/0/mathsec_lection12-message-integrity-authentication.pdf?forcedownload=1)