Отчет по лабораторной работе №7

Дисциплина: Математические основы защиты информации и информационной безопасности

Живцова Анна

Содержание

# 1 Цель работы

Изучить алгоритм Полларда для дискретного логарифмирования в конечном поле.

# 2 Задание

Реализовать алгоритм Полларда для дискретного логарифмирования в конечном поле.

# 3 Теоретическое введение

Задача дискретного логарифмирования в конечном поле – одна из первых задач, использующихся для построения криптосистем с открытым ключем. Эта задача также используется для установления сеансового ключа. Криптоскойкость данных схем основывается на вычислительной сложности решения задачи дискретного логарифмирования. Подробнее в источниках [1,2].

В данной работе будем использовать -метод Полларда, позволяющий решить задачу дискретного логарифмирования в конечном поле порядка , т.е. для нахождения такого, что . Для реализации метода нужно задать сжимающую функцию на конечном множестве. Также требуется, чтобы сохранялась возможность вычислить по известному значению и неизвестному В качестве примера такой функции в данной работе используется кусочная функция

# 4 Выполнение лабораторной работы

## 4.1 Программная реализация

Для реализации алгоритма дискретного логарифмирования методом Полладра на языке Python была написанна следующая функция.

def disk\_log(p, a, b, f, u, v):  
 c = ((a\*\*u)\*(b\*\*v))%p  
 c\_log = [u, v]  
 d = c  
 d\_log = [u, v]  
 while True:  
 print(c%p, c\_log, d%p, d\_log)  
 c, c\_log\_n = func(a, b, c, p)  
 c\_log += c\_log\_n  
 d, d\_log\_n = func(a, b, d, p)  
 d\_log += d\_log\_n  
 d, d\_log\_n = func(a, b, d, p)  
 d\_log += d\_log\_n  
 if (c-d)%p == 0:  
 order = find\_order(a, p)  
 cd\_log = c\_log - d\_log  
 r = (order - abs(cd\_log[0]))  
 for i in range(order):  
 if (r + i\*order)%cd\_log[1] == 0:  
 return abs((r + i\*order)//cd\_log[1])

Тут – порядок поля, – из условия задачи логарифмирования, – сжимающая функция, – начальные приближения.

Дополнительно были реализованы функции нахождения порядка элемента в поле и сжимающая функция на конечном множестве

def find\_order(n, p):  
 for i in range(1, p):  
 if (n\*\*(i))%p == 1:  
 return i  
  
def func(a, b, x, p):  
 if x < p/2:  
 return (a\*x)%p, np.array([1, 0])  
 return (b\*x)%p, np.array([0, 1])

## 4.2 Проверка функциональности программы

Функциональность данной функции была протестирована в среде jupyter notebook (см. рис. 1). Функция действительно помогла решить задачу дискретного логарифмирования.

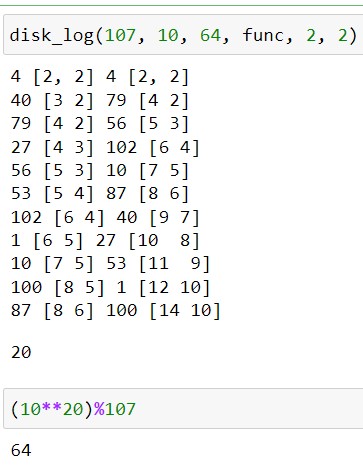


Рис. 1: Тестирование алгоритма Полладра

# 5 Выводы

В данной работе я изучила алгоритм Полладра для решения задачи дискретного логарифмирования в конечном поле. Также я реализовала его программно и протестировала.

# Список литературы

1. Kulyabov D., Korolkova A., Gevorkyan M. Информационная безопасность компьютерных сетей: лабораторные работы. 2015.

2. Самуйлов К.Е. и др. Сети и телекоммуникации : Учебник и практикум. Издательство Юрайт, 2019. С. 1–363.