Отчёт по лабораторной работе №7. Дискретное логарифмирование в конечном поле

Дисциплина: Математические основы защиты информации и информационной безопасности

Манаева Варвара Евгеньевна

Содержание

# 1 Общая информация о задании лабораторной работы

## 1.1 Цель работы

Ознакомиться с алгоритмом дискретного логарифмирования в конечном поле.

## 1.2 Задание [1]

1. Реализовать алгоритм дискретного логарифмирования в конечном поле;
2. Вычислить логарифм с заданными числами .

# 2 Теоретическое введение [2]

Алгоритм Полларда — это эффективный метод для нахождения дискретного логарифма, который использует идею случайных блужданий и метод “кролика и черепахи”.

## 2.1 Определение задачи

Дискретный логарифм — это задача нахождения целого числа x по заданным элементам g и y в конечной группе G, такой что:

где g — основание, y — результат возведения в степень, а p — простое число, определяющее порядок группы.

## 2.2 Основная идея алгоритма

Алгоритм Полларда основан на методе случайных блужданий и использует принцип “кролика и черепахи” (или “метод Флойда”). Он предполагает, что мы можем генерировать последовательности значений с помощью случайных блужданий и сравнивать их для нахождения совпадений.

Псевдокод, описывающий работу алгоритма:

input: a: a generator of G  
 b: an element of G  
output: An integer x such that a^x = b, or failure  
  
Initialise i ← 0, a\_0 ← 0, b\_0 ← 0, x\_0 ← 1 ∈ G  
  
loop  
 i ← i + 1  
  
 x\_i ← f(x\_i−1),  
 a\_i ← g(x\_i−1, a\_i−1),  
 b\_i ← h(x\_i−1, b\_i−1)  
  
 x\_2i−1 ← f(x\_2i−2),  
 a\_2i−1 ← g(x\_2i−2, a\_2i−2),  
 b\_2i−1 ← h(x\_2i−2, b\_2i−2)  
 x\_2i ← f(x\_2i−1),  
 a\_2i ← g(x\_2i−1, a\_2i−1),  
 b\_2i ← h(x\_2i−1, b\_2i−1)  
while x\_i ≠ x\_2i  
  
r ← b\_i − b\_2i  
if r = 0 return failure  
return r−1(a\_2i − a\_i) mod n

Алгоритм Полларда является мощным инструментом для решения задачи дискретного логарифмирования в конечных группах. Его эффективность в основном зависит от выбора начальных значений и структуры группы. Этот алгоритм часто используется в криптографии, особенно в контексте систем на основе эллиптических кривых и других методов шифрования.

# 3 Выполнение лабораторной работы [1]

## 3.1 Реализовать алгоритм дискретного логарифмирования в конечном поле

Исходный код написан на языке Julia [3]. Код функции, осуществляющей алгоритм дискретного логарифмирования в конечном поле, представлен ниже.

function new\_xab(x, a, b, p, alph, bett)  
 if x % 3 == 0  
 return x^2 % p, a\*2 % (p-1), b\*2 % (p-1)  
 elseif x % 3 == 1  
 return x\*alph % p, (a+1) % (p-1), b  
 else  
 return x\*bett % p, a, (b+1) % (p-1)  
 end  
end  
  
function searching\_for\_gamma(a\_diff, b\_diff, p)  
 for i in 1:p  
 if b\_diff\*i % p == a\_diff  
 return i  
 end  
 end  
 return "Not found"  
end  
  
function metodPollarda(p, alp, bet)# , any\_func::Function)  
 if p % 2 == 0  
 return "Incorrect input: p must be simple"  
 end  
 a\_i = 0; b\_i = 0; x\_i = 1  
 a\_2i = 0; b\_2i = 0; x\_2i = 1  
 i = 1  
 tries = 1000  
 data = zeros(Int64, (3, tries))  
 data2 = zeros(Int64, (3, tries))  
 while i <= tries  
 x\_i, a\_i, b\_i = new\_xab(x\_i, a\_i, b\_i, p, alp, bet)  
 data[:, i] = [x\_i, a\_i, b\_i]  
   
 x\_2i, a\_2i, b\_2i = new\_xab(x\_2i, a\_2i, b\_2i, p, alp, bet)  
 x\_2i, a\_2i, b\_2i = new\_xab(x\_2i, a\_2i, b\_2i, p, alp, bet)  
 data2[:, i] = [x\_2i, a\_2i, b\_2i]  
  
 if x\_i == x\_2i  
 display(data[:, 1:i])  
 display(data2[:, 1:i])  
 r = b\_2i - b\_i  
 if r == 0  
 return "Не найдено"  
 else  
 return searching\_for\_gamma(a\_i - a\_2i, r, p)  
 end  
 end  
 i += 1  
 end  
 return "Делитель не найден"  
end

### 3.1.1 Проверка работы функции

Для проверки работы функции проверим работу функции на лёгких значениях:

p = 1019  
alp = 2  
bet = 5  
metodPollarda(p, alp, bet)

Результат работы кода представлен ниже (рис. 1).

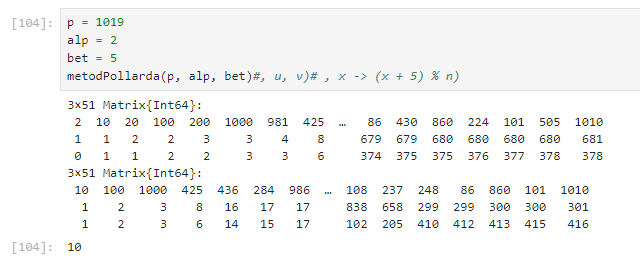


Рис. 1: Результат работы реализованной функции разложения числа на множители

## 3.2 Вычисление логарифма с заданными числами

p = 107  
alp = 10  
bet = 64  
metodPollarda(p, alp, bet)

Результат работы кода представлен ниже (рис. 2).

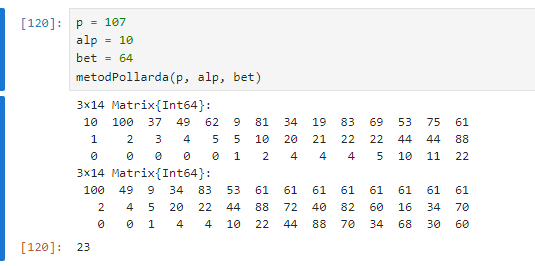


Рис. 2: Результат работы реализованной функции разложения числа на множители

# 4 Выводы

В результате работы мы ознакомились с алгоритмом дискретного логарифмирования в конечном поле и реализовали его на языке программирования Julia.

Также были записаны скринкасты:

На RuTube:

* [Весь плейлист](https://rutube.ru/plst/540770)
* [Запись создания шаблона отчёта и презентации для заполнения](https://rutube.ru/video/f2eff0bf79aae34ebe62602bdb92a9b8)
* [Выполнения лабораторной работы](https://rutube.ru/video/be5076520a95bc8f7a83720adba662ff)
* [Запись создания отчёта](https://rutube.ru/video/991104f7db162156d499eb7572878631)
* [Запись создания презентации](https://rutube.ru/video/28ab193ab31144b67f5984e370392282)
* [Защита лабораторной работы](https://rutube.ru/video/317a3461aefa7f67121e6b61a7c400be)

На Платформе:

* [Весь плейлист](https://plvideo.ru/playlist?list=vaNN02mO97J6)
* [Запись создания шаблона отчёта и презентации для заполнения](https://plvideo.ru/watch?v=xAma7VEEbvb-)
* [Выполнения лабораторной работы](https://plvideo.ru/watch?v=1kSw7LKQhOl-)
* [Запись создания отчёта](https://plvideo.ru/watch?v=RR6a43t3hZEV)
* [Запись создания презентации](https://plvideo.ru/watch?v=Ailpu5FuSQ2C)
* [Защита лабораторной работы](https://plvideo.ru/watch?v=0Iq7_OQGMLgB)

# Список литературы

1. Лабораторная работа №7. Дискретное логарифмирование в конечном поле [Электронный ресурс]. RUDN, 2024. URL: <https://esystem.rudn.ru/pluginfile.php/2368518/mod_folder/content/0/lab07.pdf>.

2. Математика криптографии и теория шифрования [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/552/408/info>.

3. Julia 1.10 Documentation [Электронный ресурс]. 2024. URL: <https://docs.julialang.org/en/v1/>.