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

Дисциплина: Математические основы защиты информации и информационной безопасности

Манаева Варвара Евгеньевна

Содержание

Список иллюстраций

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 p, a, b.

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:

$$g^x \equiv y(mod p)$$

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"
```

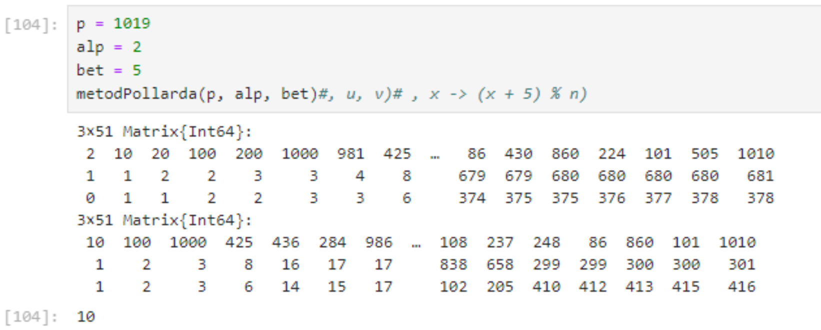
```
            else
                return searching_for_gamma(a_i - a_2i, r, p)
            end
        end
        i += 1
    end
    return "Делитель не найден"
end
```

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. 3.1).



```
[104]: p = 1019
       alp = 2
       bet = 5
       metodPollarda(p, alp, bet)#, u, v)# , x -> (x + 5) % n)

       3×51 Matrix{Int64}:
        2  10  20  100  200  1000  981  425  …   86  430  860  224  101  505  1010
        1   1   2    2    3     3    4    8     679  679  680  680  680  680   681
        0   1   1    2    2     3    3    6     374  375  375  376  377  378   378
       3×51 Matrix{Int64}:
        10  100  1000  425  436  284  986  …  108  237  248   86  860  101  1010
         1    2     3    8   16   17   17     838  658  299  299  300  300   301
         1    2     3    6   14   15   17     102  205  410  412  413  415   416

[104]: 10
```

Рис. 3.1: Результат работы реализованной функции разложения числа на множители

3.2 Вычисление логарифма с заданными числами p, a, b

```
p = 107

alp = 10

bet = 64

metodPollarda(p, alp, bet)
```

Результат работы кода представлен ниже (рис. 3.2).

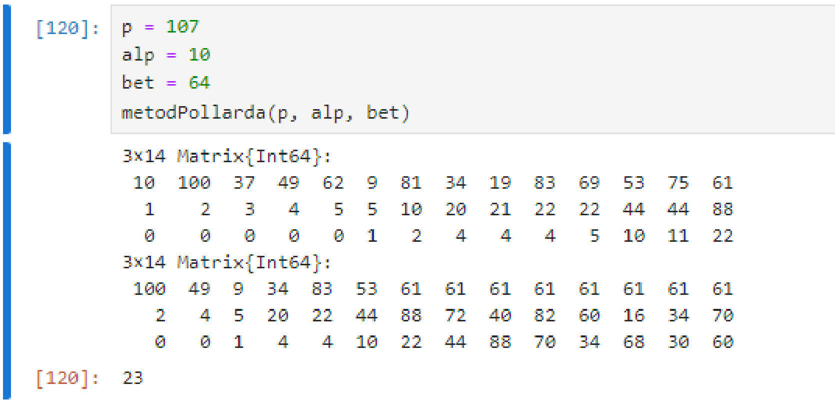


Рис. 3.2: Результат работы реализованной функции разложения числа на множители

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
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

На Платформе:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

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/.