Лабораторная работа №7

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

4 декабря 2024

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

Прагматика выполнения

Прагматика выполнения

- Освоение алгоритмов дискретного логарифмирования в конечном поле — ρ-метода Полларда для дискретного логарифмирования.

Цели

Цели

Изучить работу алгоритмов дискретного логарифмирования в конечном поле — ρ-метод Полларда для дискретного логарифмирования, а также реализовать его программно.

Задачи

Задачи

1. Реализовать алгоритм программно для некоторых чисел p, a, b и вычислить логарифм.

Выполнение работы

ρ-Метода Полларда (1/4)

C: > Users > User > Documents > work > study > 2024-2025 > Математичe

```
using LinearAlgebra ✓

function f_c(a::Int, b::Int, c::Int, p::Int)
    if c < div(p, 2)
        return (a * c) % p, [1, 0]
    else
        return (b * c) % p, [0, 1]
    end
end

function Find_Order(n::Int, p::Int)
    if n % p == 1
        return 1
    end
    t = n % p
    for i in 2:p
        if (t * n) % p == 1
            return i
        end
        t = (t * n) % p
    end
    return nothing
end

function Power_Mod(a::Int, b::Int, p::Int)::Int
    res = 1
    a = a % p
    while b > 0
        if b % 2 == 1
            res = (res * a) % p
        end
        b = div(b, 2)
        a = (a * a) % p
    end
    return res
end
```

ρ-Метода Полларда (2/4)

```julia
function Pollard_Rho_Method(p::Int, a::Int, b::Int, u = 2, v = 2)
    c = (a^u * b^v) % p
    c_log = [u, v]
    d = c
    d_log = [u, v]

    i = 0
    while true
        println("Шаг: $i, \t c: $(c % p), c_log: $(c_log[2])x + $(c_log[1]), \t d: $(d % p), d_log: $(d_log[2])x + $(d_log[1])")
        i += 1
        # Обновим значения для c
        c, c_log_n = f_c(a, b, c, p)
        c_log += c_log_n
        # Обновим значения для d (двойной шаг)
        d, d_log_n = f_c(a, b, d, p)
        d_log += d_log_n
        d, d_log_n = f_c(a, b, d, p)
        d_log += d_log_n

        # Проверяем совпадение c и d
        if (c - d) % p == 0
            println("Шаг: $i, \t c: $(c % p), c_log: $(c_log[2])x + $(c_log[1]), \t d: $(d % p), d_log: $(d_log[2])x + $(d_log[1])")
            order = Find_Order(a, p)
            if order === nothing
                return "Порядок числа a = $a (по модулю p = $p) не найден"
            end

            cd_log = c_log - d_log
            r = (order - abs(cd_log[1]))
            for i in 0:(order - 1)
                if (r + i * order) % cd_log[2] == 0
                    return abs(div(r + i * order, cd_log[2]))
                end
            end
        end
    end
end
```

Рис. 2: Код алгоритма ρ-Метода Полларда для дискретного логарифмирования на Julia

ρ-Метода Полларда (3/4)

```
# Исходные данные a^b = x (mod p)
a1 = 10 # Основание логарифма
b1 = 64 # Число, для которого ищем логарифм
p1 = 107 # Модуль конечного поля
println("Начальные данные: a = $a1, b = $b1, p = $p1, r = $(Find_Order(a1, p1))")

# Вызов функции p-метода Полларда для дискретного логарифмирования
x1 = Pollard_Rho_Method(p1, a1, b1)

println("Дискретный логарифм 10^64 = x (mod 107) равняется: x = $x1")

# Проверка результата
if Power_Mod(a1, x1, p1) == b1
    println("Дискретный логарифм: x = $x1")
else
    println("Решение не найдено")
end
println("\n")


# Исходные данные a^b = x (mod p)
a2 = 2 # Основание логарифма
b2 = 22 # Число, для которого ищем логарифм
p2 = 29 # Модуль конечного поля
println("Начальные данные: a = $a2, b = $b2, p = $p2, r = $(Find_Order(a2, p2))")

# Вызов функции p-метода Полларда для дискретного логарифмирования
x2 = Pollard_Rho_Method(p2, a2, b2)

println("Дискретный логарифм 2^22 = x (mod 29) равняется: x = $x2")

# Проверка результата
if Power_Mod(a2, x2, p2) == b2
    println("Дискретный логарифм: x = $x2")
else
    println("Решение не найдено")
end
```

ρ-Метода Полларда (4/4)

```
thbase-infosec\labs\lab07\report\report> julia .\lab7.jl
Начальные данные: a = 10, b = 64, p = 107, r = 53
Шаг: 0,          c: 4, c_log: 2x + 2,    d: 4, d_log: 2x + 2
Шаг: 1,          c: 40, c_log: 2x + 3,   d: 79, d_log: 2x + 4
Шаг: 2,          c: 79, c_log: 2x + 4,   d: 56, d_log: 3x + 5
Шаг: 3,          c: 27, c_log: 3x + 4,   d: 75, d_log: 5x + 5
Шаг: 4,          c: 56, c_log: 3x + 5,   d: 3, d_log: 7x + 5
Шаг: 5,          c: 53, c_log: 4x + 5,   d: 86, d_log: 7x + 7
Шаг: 6,          c: 75, c_log: 5x + 5,   d: 42, d_log: 8x + 8
Шаг: 7,          c: 92, c_log: 6x + 5,   d: 23, d_log: 9x + 9
Шаг: 8,          c: 3, c_log: 7x + 5,    d: 53, d_log: 9x + 11
Шаг: 9,          c: 30, c_log: 7x + 6,   d: 92, d_log: 11x + 11
Шаг: 10,         c: 86, c_log: 7x + 7,   d: 30, d_log: 12x + 12
Шаг: 11,         c: 47, c_log: 8x + 7,   d: 47, d_log: 13x + 13
Дискретный логарифм 10^64 = x (mod 107) равняется: x = 20
Дискретный логарифм: x = 20


Начальные данные: a = 2, b = 22, p = 29, r = 28
Шаг: 0,          c: 22, c_log: 2x + 2,   d: 22, d_log: 2x + 2
Шаг: 1,          c: 20, c_log: 3x + 2,   d: 5, d_log: 4x + 2
Шаг: 2,          c: 5, c_log: 4x + 2,    d: 20, d_log: 4x + 4
Шаг: 3,          c: 10, c_log: 4x + 3,   d: 10, d_log: 5x + 5
Дискретный логарифм 2^22 = x (mod 29) равняется: x = 26
Дискретный логарифм: x = 26
```

Рис. 4: Результат выполнения кода по решению задачи дискретного логарифмирования на Julia

Результаты

Результаты

По результатам работы, я изучил работу алгоритмов дискретного логарифмирования в конечном поле — ρ-метода Полларда для дискретного логарифмирования, а также реализовал его программно.