Лабораторная работа №7

Математические основы защиты информации и информационной безопасности

Данилова Анастасия Сергеевна

Содержание

Цель работы 1
Задание 1
Теоретическое введение 1
Выполнение лабораторной работы 2
Выводы 3
Список литературы 3

Цель работы

Изучить дискретное логарифмирование в конечном поле и реализовать алгоритм программно на языке программмирования Julia.

Задание

- Изучить теоретическую часть о дискретном логарифмировании;
- Реализовать алгоритм программно.

Теоретическое введение

Р-метод Полларда

Ро-алгоритм — предложенный Джоном Поллардом в 1975 году алгоритм, служащий для факторизации (разложения на множители) целых чисел. Данный алгоритм основывается на алгоритме Флойда поиска длины цикла в последовательности и некоторых следствиях из парадокса дней рождения. Алгоритм наиболее эффективен при факторизации составных чисел с достаточно малыми множителями в разложении.

Дискретное логарифмирование

Дискретное логарифмирование (DLOG) — задача обращения функции g^x в некоторой конечной мультипликативной группе G.

Наиболее часто задачу дискретного логарифмирования рассматривают в мультипликативной группе кольца вычетов или конечного поля, а также в группе точек эллиптической кривой над конечным полем. Эффективные алгоритмы для решения задачи дискретного логарифмирования в общем случае неизвестны.

Для заданных g и a решение x уравнения g^x = a называется дискретным логарифмом элемента a по основанию g. В случае, когда G является мультипликативной группой кольца вычетов по модулю m, решение называют также индексом числа a по основанию g. Индекс числа a по основанию g гарантированно существует, если g является первообразным корнем по модулю m.

Выполнение лабораторной работы

```
using Random

function mod_exp(base, exp, mod)
    result = 1
    base = base % mod
    while exp > 0
        if exp % 2 == 1
            result = (result * base) % mod
        end
        exp = div(exp, 2)
        base = (base * base) % mod
    end
    return result
end

function pollard(g, h, p)
    function f(x)
        return (mod_exp(g, x[1], p) * mod_exp(g, x[2], p) % p, (x[1] + 1) % (p - 1), (x[2] + 1) % (p - 1))
    end

    x0 = (0, 0)
    x1 = f(x0)
    x2 = f(f(x0))

```

Код

```
    while true
        if mod_exp(g, x1[1], p) == h
            return x1[1]
        end
        if mod_exp(g, x2[1], p) == h
            return x2[1]
        end

        if x1 == x2
            return "Не удалось найти логарифм"
        end

        x1 = f(x1)
        x2 = f(f(x2))
    end
end

println("g: ")
g = parse(Int, readline())
println("h: ")
h = parse(Int, readline())
println("p: ")
p = parse(Int, readline())
result = pollard(g, h, p)
println("Логарифм: ", result)
```

Код2

$$3^x \equiv 13 \pmod{17}.$$

Пример задачи

```
  Activating new project at `C:\Users\nastd\.julia\environments\v1.11`
g:
3
h:
13
p:
17
Логарифм: 4
  * Terminal will be reused by tasks, press any key to close it.
```

Результат

Выводы

Мы изучили теоретическую часть о дискретном логарифмировании в конечном поле и реализовали алгоритм на языке программмирования Julia.

Список литературы

1. Mathematics // Julia URL: https://docs.julialang.org/en/v1/base/math/