Лабораторная работа №7: Дискретное логарифмирование в конечном поле

Реализация p-метода Полларда

Студент: Назарова Дарья Владиславовна

Группа: НПИмд01-24

Содержание

Теоретические основы Постановка задачи Алгоритм Полларда Реализация на Julia Примеры работы Выводы Приложения

1. Теоретические основы

Конечные поля GF(p)

Определение: Множество $F_p = Z / p Z$ классов вычетов по модулю простого числа $p$ является конечным полем из $p$ элементов.

Свойства:

Существуют операции сложения и умножения Каждый ненулевой элемент имеет обратный Мультипликативная группа циклическая Существуют генераторы (первообразные корни) Дискретный логарифм

Задача: Для заданных $a$, $b$ и простого $p$ найти $x$ такой, что:

a x ≡ b ( m o d p ) a x ≡b(modp) Обозначение: $x = \log_a b \left( mod\ p \right)$

Сложность:

Вычисление $a^x\ mod\ p$: полиномиальная сложность Нахождение $x$ по $a^x\ mod\ p$: субэкспоненциальная сложность

1. Постановка задачи

Цель работы

Изучить задачу дискретного логарифмирования в конечных полях Реализовать p-метод Полларда Применить алгоритм для решения практических задач Исходные данные

Простое число $p$ Основание $a$ (генератор мультипликативной группы) Число $b$, для которого ищется логарифм Ожидаемый результат

Значение $x$, удовлетворяющее сравнению $a^x \equiv b \left( mod\ p \right)$ Программная реализация алгоритма

1. Алгоритм Полларда

Основная идея

Метод "черепахи и зайца" (Floyd's cycle-finding):

Использует псевдослучайное блуждание Обнаруживает коллизии за $O(\sqrt{n})$ шагов Требует константной памяти Математическая модель

Ветвящееся отображение:

f ( c ) = { a ⋅ c ( m o d p ) , если c < p 2 b ⋅ c ( m o d p ) , если c ≥ p 2

f(c)={ a⋅c(modp), b⋅c(modp),

если c< 2 p

если c≥ 2 p

Свойства отображения:

Сжимающее: уменьшает пространство поиска Вычислимое: позволяет отслеживать логарифмы Детерминированное: гарантирует повторяемость Шаги алгоритма

Шаг 1: Инициализация

julia u, v ← случайные числа из [0, r-1] c ← a^u * b^v mod p d ← c α_c, β_c ← u, v # log(c) = α + β*x α_d, β_d ← u, v Шаг 2: Поиск коллизии

julia while c ≠ d: c, α_c, β_c ← f(c, α_c, β_c) # один шаг d, α_d, β_d ← f(d, α_d, β_d) # два шага d, α_d, β_d ← f(d, α_d, β_d) Шаг 3: Решение уравнения

При $c = d$ получаем:

α c

- β c x ≡ α d
- β d x ( m o d r ) α c  +β c  x≡α d  +β d  x(modr) Преобразуем к виду:

( β c − β d ) x ≡ α d − α c ( m o d r ) (β c  −β d  )x≡α d  −α c  (modr) Решаем линейное сравнение:

A x ≡ B ( m o d r ) Ax≡B(modr)