Лабораторная работа №7

Дискретное логарифмирование в конечном виде

- Кюнкриков Даниил Саналович
- студент уч. группы НПИмд-01-24
- Российский университет дружбы народов
- 1132249574@pfur.ru
- https://github.com/DanzanK/2025-2026_math-sec/tree/main

Объект и предмет исследования

- Объект исследования - алгоритмы дискретного логарифмированияв конечном поле
- Предмет исследования - p - метод Полларда для дискретного логарифмирования, задачи дискретного логарифмирования в конечных полях, конечные поля, метод черепах и зайца для поиска коллизий

Цели и задачи

- Цель работы: программная релизация p - метода Полларда для дискретного логарифмирования задачи:
- Задачи:
  - Разобрать постановку DLP (задачи дискретного логарифмирования)
  - Изучить принцип ρ-метода Полларда: построение последовательности, поиск коллизии и восстановление
  - Реализовать необходимые вычисления: возведение в степень по модулю, НОД/ расширенный Евклид, модульная инверсия.
  - Программно реализовать алгоритм
  - провести тестирования алгоритма в зависимости от заданных параметров поля

Теоретическая часть

Задача дикретного логарифмирования (DLP)

Для заданных простого числа (p), основания (a) (образующего элемента мультипликативной группы (_p^*)) и числа (b) требуется найти целое число (x), при котором

[a^x b (p)]

где (1 < a < p), (1 < b < p)

Конечные поля

Множество классов вычетов по модулю простого числа (p) образует конечное поле (_p = /p), которое обладает следующими свойствами: - Сложение и умножение определены по модулю (p) - Существует мультипликативная группа (_p^*) порядка (p-1) - Каждый ненулевой элемент имеет обратный по умножению

Процесс выполнения работы

Реализация метода Полларда для задачи дискретного логарифмирования используя язык программирования python

```
import secrets
import math
# Расширенный алгоритм Евклида
def egcd(a: int, b:int):
    if b == 0:
        return (a,1,0)
    g, y1, x1 = egcd(b, a % b)
    return (g, y1, x1 - (a//b) * y1)
def rho_dlog(p:int,a:int,b:int,r:int):
    # строим псевдослучайную последовательность в группе и ищем коллизию c == d,
    # Случайная инициализация логарифмов (u, v) в диапазоне [0, r-1]
    u = secrets.randbelow(r)
    v = secrets.randbelow(r)
    # Стартовая точка:
    c = (pow(a, u, p) * pow(b,v,p)) % p
    # Для поиска коллизий используем "черепаха-заяц" (Floyd):
    d = c
    alpha1, beta1, alpha2, beta2 = u, v, u, v
    # Простое разбиение множества состояний на 2 части
    half = p//2

    def step(x:int):
        if x < half:
            return(a*x) % p
        else:
            return(b*x) % p
    # Основной цикл: идём, пока не найдём коллизию c == d
    while True:
        c = step(c)
        alpha1 = (alpha1 + (1 if c < half else 0)) % r
        beta1 = (beta1 + (1 if c >= half else 0)) % r
        d = step(d)
        d = step(d)
        alpha2 = (alpha2 + 2 * (1 if d < half else 0)) % r
        beta2 = (beta2 + 2 * (1 if d >= half else 0)) % r
```

```
        if c == d:
            break
        A = (beta1 - beta2) % r
        delta = (alpha2 - alpha1) % r
        # Решаем A*x ≡ delta (mod r)
        # Если g = gcd(A, r) не делит delta — решения нет (Invalid problem)
        g, xcoef, _ = egcd(A,r)
        if delta % g !=0:
            return None
        return (xcoef * (delta // g)) % r
```

Код вызова алгоритма задачи дискретного логарифмирования

```
if __name__ == "__main__":
    while True:
        print(" Введите p, a, b, r через пробел, ВЫХОД для того чтобы выйти")
        s = input().strip().lower()
        if s == "выход":
            break
        try:
            parts = s.split()
            if len(parts) != 4:
                raise ValueError("ERROR")
            p, a, b, r = map(int, parts)
            x = rho_dlog(p, a, b, r)
            print("Invalid problem" if x is None else f"x = {x}")
        except Exception:
            print("Error. Введите p, a, b, r через пробел, ВЫХОД для того чтобы выйти ")
```

Результат работы программы с заданными параметрами: p = 107, a = 10, b = 64

```
Введите p, a, b, r через пробел, ВЫХОД для того чтобы выйти
107 10 64 53
x = 2
 Введите p, a, b, r через пробел, ВЫХОД для того чтобы выйти
107 10 64 13
x = 12
```

Вывод

Реализован p-метод Полларда по дискретному логарифмированию в конечном поле на языке программирования python. Также вычислен логарифм на основе заданных параметрах конечного поля