Лабораторная работа 7

Попов Дмитрий Павлович, НФИмд-01-23

Содержание

List of Figures

List of Tables

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7

дисциплина: Математические основы защиты информации и информационной безопасности

Преподователь: Кулябов Дмитрий Сергеевич

Студент: Попов Дмитрий Павлович

Группа: НФИмд-01-23

МОСКВА

2023 г.

1 Цель работы

Освоить на практике дискретное логарифмирование в конечном поле.[1]

2 Выполнение лабораторной работы

Требуется реализовать:

1. Алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования

2.1 p-метод Полларда

Основные шаги:

Вход: Простое число p, числа a порядка r по модулю p, целое число b, 1< b < p отображение f, обладающее сжимающими свойствами и сохраняющее вычислимость логарифма Выход: Показатель x, Для которого a^x Тождественно = b (mod p), если такой показатель существует 1. Выбрать произвольные числа u, v и положить c <- a^u * b^v (mod p), d <- c 2. Выполнять c <- f(c)(mod p), d <- f(f(d))(mod p), вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения равенства c тождественно = d(mod p) 3. Приравняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат: x или "Решения нет"

Чтобы реализовать программу был написал след. код на python:

1. Функция, реализующая p-метод Полларда fig. 2.1.
2. Функция нахождения НОД fig. 2.2.
3. Расширенный алгоритм Евклида для вычисления модульного обратного элемента fig. 2.3.

```python
def pollard_p_method(p, a, b, f, r, u, v):
    # Выбор произвольных чисел u, v
    c = (a ** u * b ** v) % p
    d = c

    # Итерации метода Полларда
    while True:
        c = f(c) % p
        d = f(f(d)) % p

        # Если c = d, вычисляем логарифмы для c и d
        if c == d:
            # Вычисляем логарифм x решением сравнения по модулю r
            # Решения нет, если r и p не взаимно просты
            if gcd(r, p - 1) != 1:
                return "Решения нет"

            # Вычисляем логарифм x
            x = (u - v * modinv((u - v), r) * (c - a ** u) % r) % r
            return x
```

Figure 2.1: pollard

```python
# Функция вычисления наибольшего общего делителя (GCD)
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
```

Figure 2.2: gcd

```python
# Расширенный алгоритм Евклида для вычисления модульного обратного элемента
def modinv(a, m):
    m0, x0, x1 = m, 0, 1
    while a > 1:
        q = a // m
        m, a = a % m, m
        x0, x1 = x1 - q * x0, x0
    return x1 + m0 if x1 < 0 else x1
```

Figure 2.3: modinv

Выходные значения программы fig. 2.4.

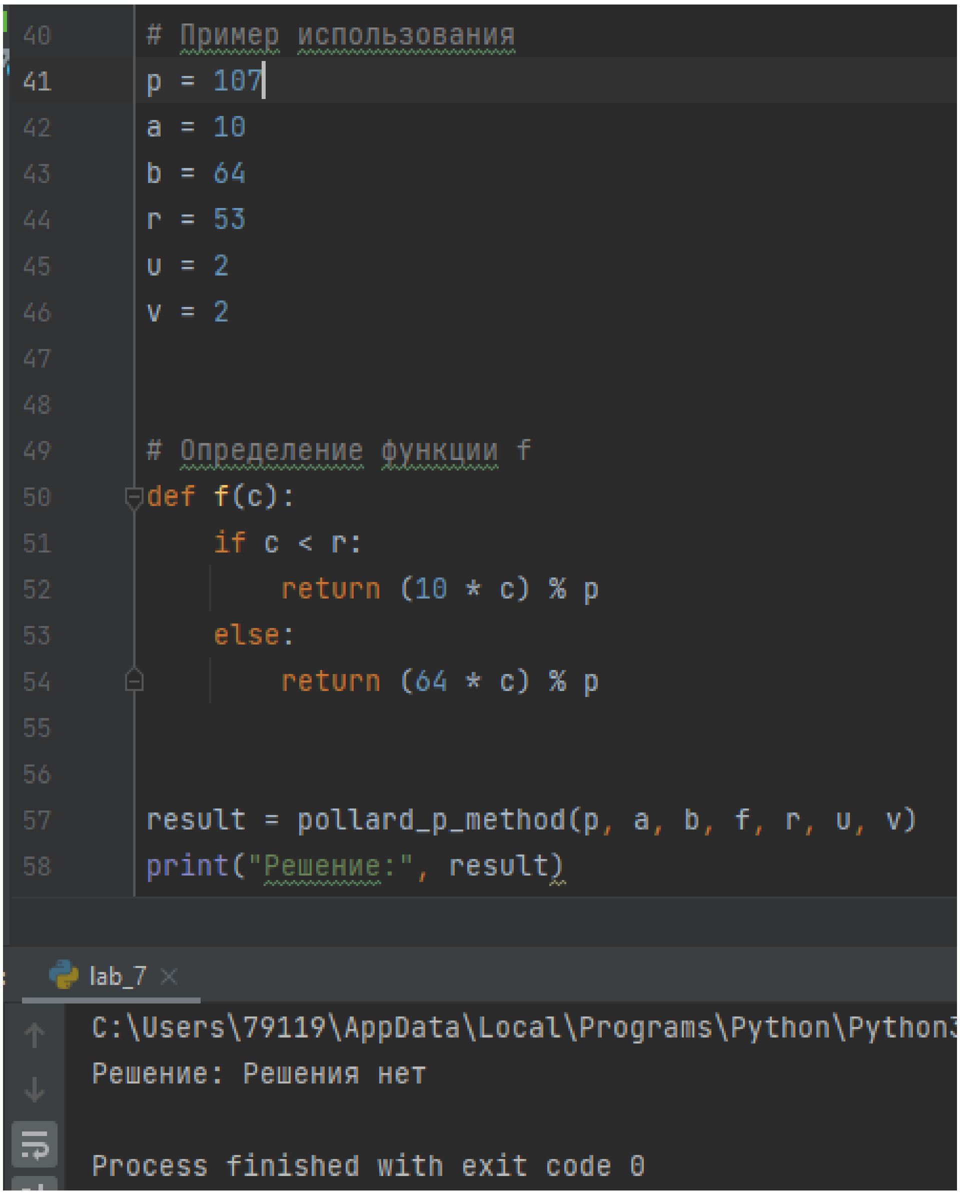

Figure 2.4: output

3 Выводы

В результате выполнения работы я освоил на практике дискретное логарифмирование в конечном поле.

4 Список литературы

1. Методические материалы курса