

# **Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе №7**

**Шифрование гаммированием**

Терентьев Егор Дмитриевич 1132236902

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	р-метод Полларда . . . . .	6
<b>3</b>	<b>Выводы</b>	<b>9</b>
<b>4</b>	<b>Список литературы</b>	<b>10</b>

# List of Figures

2.1	main_func . . . . .	7
2.2	output . . . . .	8

## List of Tables

# 1 Цель работы

Освоить на практике дискретное логарифмирование в конечном поле.

## 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  $a^x \equiv b \pmod{p}$ , если такой показатель существует  
1. Выбрать произвольные числа  $u, v$  и положить  $s \leftarrow a^u \cdot b^v \pmod{p}$ ,  $d \leftarrow s$   
2. Выполнять  $s \leftarrow f(s) \pmod{p}$ ,  $d \leftarrow f(f(d)) \pmod{p}$ , вычисляя при этом логарифмы для  $s$  и  $d$  как линейные функции от  $x$  по модулю  $r$ , до получения равенства  $s \equiv d \pmod{p}$   
3. Приравняв логарифмы для  $s$  и  $d$ , вычислить логарифм  $x$  решением сравнения по модулю  $r$ .  
Результат:  $x$  или “Решения нет”

Чтобы реализовать программу был написан след. код на python:

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

```

1  def pollard_p_method(p, a, b, f, r, u, v):
2      # Выбор произвольных чисел u, v
3      c = (a ** u * b ** v) % p
4      d = c
5
6      # Итерации метода Полларда
7      while True:
8          c = f(c) % p
9          d = f(f(d)) % p
10
11         # Если c = d, вычисляем логарифмы для c и d
12         if c == d:
13             # Вычисляем логарифм x решением сравнения по модулю r
14             # Решения нет, если r и p не взаимно просты
15             if gcd(r, p-1) != 1:
16                 return "Решения нет"
17
18             # Вычисляем логарифм x
19             x = (u - v * modinv((u - v), r) * (c - a ** u) % r) % r
20             return x
21
22     # Функция вычисления наибольшего общего делителя (GCD)
23     def gcd(a, b):
24         while b:
25             a, b = b, a % b
26         return a
27
28     # Расширенный алгоритм Евклида для вычисления модульного обратного элемента
29     def modinv(a, m):
30         m0, x0, x1 = m, 0, 1
31         while a > 1:
32             q = a // m
33             m, a = a % m, m
34             x0, x1 = x1 - q * x0, x0
35         return x1 + m0 if x1 < 0 else x1
36

```

Figure 2.1: main\_func

Выходные значения программы fig. 2.2.

```
37 # Пример использования
38 p = 107
39 a = 10
40 b = 64
41 r = 53
42 u = 2
43 v = 2
44
45 # Определение функции f
46 def f(c):
47     if c < r:
48         return (10 * c) % p
49     else:
50         return (64 * c) % p
51
52 result = pollard_p_method(p, a, b, f, r, u, v)
53 print("Решение:", result)
54
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS F:\учеба 5 курс\информационная безопасность> python .\Lab7\_log\lab7\_logs.py  
Решение: Решения нет
- PS F:\учеба 5 курс\информационная безопасность> █

Figure 2.2: output



## 3 Выводы

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

## **4 Список литературы**

1. Методические материалы курса