Лабораторная работа №7

Математические основы защиты информации и информационной безопасности

Леонтьева К. А., НПМмд-02-23

5 ноября 2023

Российский университет дружбы народов

Москва, Россия

Цель лабораторной работы

1) Реализовать на языке программирования p-метод Полларда для дискретного логарифмирования

Обозначим $F_p = Z/pZ, p$ - простое целое число и назовем конечным полем из p элементов. **Задача дискретного логарифмирования** в конечном поле F_p формулируется так: для данных целых чисел a и b, $a > 1, b > p$, найти логарифм - такое целое число x, что $a^x \equiv b(mod\ p)$ (если такое число существует). По аналогии с вещественными числами используется обозначение $x = log_a b$.

Безопасность соответствующих криптосистем основана на том, что зная числа a, x, p вычислить $a^x \, (mod \, p)$ легко, а решить задачу дискретного логарифмирования трудно. Рассмотрим **p-метод Полларда**, который можно применить и для задач дискретного логарифмирования. При этом случайное отображение f должно обладать не только сжимающими свойствами, но и вычислимостью логарифма (логарифм числа $f(c)$ можно выразить через неизвестный логарифм x и $log_a f(c)$). Для дискретного логарифмирования в качестве случайного отображения f чаще всего используются ветвящиеся отображения, например:

$$f(c) = \begin{cases} ac & \text{при } c < \frac{p}{2} \\ bc & \text{при } c > \frac{p}{2} \end{cases}$$

При $c < \frac{p}{2}$: $log_a f(c) = log_a c + 1$, при $c > \frac{p}{2}$: $log_a f(c) = log_a c + x$.

Ход выполнения лабораторной работы

- Реализуем p-метод Полларда для дискретного логарифмирования

```python
import numpy as np
import math

a = 10
b = 64
p = 107

def f(x, u, v):
    if x < r:
        return (a * x) % p, u + 1, v
    if x >= r:
        return (b * x) % p, u, v + 1

def r(a,p):
    r = 1
    while (a**r - 1) % p != 0:
        r = r + 1
    return r

u = 2
v = 2
r = r(a, p)

c = (a**u * b**v) % p
d = c
```

Figure 1: Рис.1: p-метод Полларда для дискретного логарифмирования

Ход выполнения лабораторной работы

```
u_c = u
u_d = u
v_c = v
v_d = v

print('  c', '    | ', 'log_a(c)', ' |   ', ' d', '   |  ', 'log_a(d)')
print('-----------------------------------------------')
print('c =', c, ' |  ', u_c, '+', v_c, 'x', ' |  ', 'd =', d, ' |  ', u_d, '+', v_d, 'x')

c, u_c, v_c = f(c, u_c, v_c)
d, u_d, v_d = f(f(d, u_d, v_d)[0], f(d, u_d, v_d)[1], f(d, u_d, v_d)[2])

print('c =', c, ' |  ', u_c, '+', v_c, 'x', ' |  ', 'd =', d, ' |  ', u_d, '+', v_d, 'x')

while c % p != d % p:
    c, u_c, v_c = f(c, u_c, v_c)
    d, u_d, v_d = f(f(d, u_d, v_d)[0], f(d, u_d, v_d)[1], f(d, u_d, v_d)[2])

    print('c =', c, ' |  ', u_c, '+', v_c, 'x', ' |  ', 'd =', d, ' |  ', u_d, '+', v_d, 'x')

x = 1
while (u_c + v_c * x) % r != (u_d + v_d * x) % r:
    x = x + 1

print(' ')
print('Показатель x =', x)
```

Figure 2: Рис.2: p-метод Полларда для дискретного логарифмирования

Ход выполнения лабораторной работы

```
  c     |  log_a(c)  |     d    |   log_a(d)
-------------------------------------------
c = 4   |   2 + 2 x  |   d = 4  |   2 + 2 x
c = 40  |   3 + 2 x  |   d = 79 |   4 + 2 x
c = 79  |   4 + 2 x  |   d = 56 |   5 + 3 x
c = 27  |   4 + 3 x  |   d = 75 |   5 + 5 x
c = 56  |   5 + 3 x  |   d = 3  |   5 + 7 x
c = 53  |   5 + 4 x  |   d = 86 |   7 + 7 x
c = 75  |   5 + 5 x  |   d = 42 |   8 + 8 x
c = 92  |   5 + 6 x  |   d = 23 |   9 + 9 x
c = 3   |   5 + 7 x  |   d = 53 |   11 + 9 x
c = 30  |   6 + 7 x  |   d = 92 |   11 + 11 x
c = 86  |   7 + 7 x  |   d = 30 |   12 + 12 x
c = 47  |   7 + 8 x  |   d = 47 |   13 + 13 x

Показатель x = 20
```

Figure 3: Рис.3: p-метод Полларда для дискретного логарифмирования

Вывод

- В ходе выполнения данной лабораторной работы был реализован p-метод Полларда для дискретного логарифмирования