Математические основы защиты информации и информационной безопасности

Супонина Анастасия Павловна

07 Декабрь 2024

РУДН, Москва, Россия

Цель работы

Ознакомиться с дискретным логарифмированием и научиться выполнять его программно при помощи p-метод Полларда.

Задание

Реализовать p-метод Полларда для дискретного логарифмирования на языке программирования Julia.

Алгоритм, реализующий p–Метод Полларда для задач дискретного логарифмирования - 1 этап.

Вход. Простое число p, число a порядка r по модулю p, целое число b, 1 < b < p; отображение f, обладающее сжимающими свойствами и сохраняющее вычислимость логарифма.

```
1   p = 107
2   a = 10
3   r = 53
4   b = 64
```

Рис. 1: Входные данные

Для c и для d значения u, v присваиваю разные для счета.

```
17  # Инициализация значений для c и d
18  u, v = 2, 2  # Для c
19  U, V = 2, 2  # Для d
20
21  c = a^u * b^v % p  # Начальное значение c
22  d = c  # Начальное значение d
```

Алгоритм - 2 этап - нахождения отображения.

Для начала отдельно создаю функцию, которая будет считать отбражение, а также увеличивать значения u, v на 1 на каждом шаге.

```
6   # Функция, определяющая преобразование
7   function otobr(t, z, w)
8       if t < r
9           z += 1
10          return mod(a * t, p), z, w
11      else
12          w += 1
13          return mod(b * t, p), z, w
14      end
15  end
```

Рис. 3: Функция отображения

Алгоритм - 2 этап - нахождение коллизии.

Таким образом ищу коллизии.

```
28  c, u, v = otobr(c, u, v)
29  d, U, V = otobr(d, U, V)
30  d, U, V = otobr(d, U, V)
31
32  println("Обновленное значение c: ", c)
33  println("Обновленное значение d: ", d)
34
35  # Функция для обнаружения коллизий
36  function second(c, d, u, v, U, V)
37      while c != d
38          c, u, v = otobr(c, u, v)  # Обновление для медленного указателя
39          d, U, V = otobr(d, U, V)  # Первое обновление для быстрого указателя
40          d, U, V = otobr(d, U, V)  # Второе обновление для быстрого указателя
41          println("Текущее значение c: $c, d: $d")
42      end
43      return c, d, u, v, U, V
44  end
45
46  # Находим коллизию
47  c, d, u, v, U, V = second(c, d, u, v, U, V)
```

Алгоритм - 3 этап - функция нахождения обратного значения.

Создаю функцию для вычисления обратного элемента, чтобы потом вычислить значение x.

```
74  # Функция для вычисления обратного элемента по модулю
75  function invmod(a, m)
76      g, x, _ = gcdx(a, m)
77      if g != 1
78          throw(ArgumentError("Обратного элемента не существует"))
79      else
80          return mod(x, m)
81      end
82  end
```

Рис. 5: обратное значение

Алгоритм - 3 этап - функция дляя решения логарифмов и нахождения значения x.

```
# Функция для вычисления x из логарифмов
function compute_x(u, v, U, V, r)
    # Вычисляем разницу логарифмов
    delta_v = mod(v - V, r)  # Теперь Δv используется с x
    delta_u = mod(U - u, r)

    if delta_v == 0
        return "Решений нет"  # Если delta_v = 0, решения не существует
    end

    delta_v_inv = invmod(delta_v, r)

    # Вычисляем x
    x = mod(delta_u * delta_v_inv, r)  # Здесь x = Δu * (Δv)^-1 mod r
    return x
end
```

Результат

```
Начальное значение c: 4
Начальное значение d: 4
Обновленное значение c: 40
Обновленное значение d: 79
Текущее значение c: 79, d: 56
Текущее значение c: 27, d: 75
Текущее значение c: 56, d: 3
Текущее значение c: 53, d: 86
Текущее значение c: 75, d: 42
Текущее значение c: 92, d: 23
Текущее значение c: 3, d: 53
Текущее значение c: 30, d: 92
Текущее значение c: 86, d: 30
Текущее значение c: 47, d: 47
Итоговое значение c: 47
Итоговое значение d: 47
Итоговое значение u: 7
Итоговое значение v: 8
Итоговое значение U: 13
Итоговое значение V: 13
Логарифм x: 20
```

Выводы

В процессе выполнения работы, я реализовала алгоритм p-Полларда для задач дискретного логарифмирования на языке программирования Julia.