Математические основы защиты информации и информационной безопасности

Нирдоши Всеволод Раджендер

07 Декабря 2024

РУДН, Москва, Россия

Лабораторная работа №7

Цель работы

Изучение задачи дискретного логарифмирования в конечных полях и алгоритма p-метода Полларда для её решения. Реализация алгоритма с использованием языка программирования Julia.

Задачи

1. Разобраться с основами конечных полей и их свойствами, включая операции сложения, умножения и нахождения обратных элементов.
2. Изучить теоретическую основу p-метода Полларда для решения задачи дискретного логарифмирования.
3. Реализовать алгоритм на языке Julia, обеспечивая корректность вычислений на каждом этапе.
4. Проверить работоспособность алгоритма на конкретных данных и проанализировать результаты.

Проделанная работа:

1. **Теоретическая часть:**
    - Изучены основные свойства конечных полей и их применение в криптографии.
    - Разобрана суть задачи дискретного логарифмирования:

      $$a^x \equiv b \pmod p,$$

      где p — простое число, a — основание, b — значение, и x — логарифм, который необходимо найти.
    - Изучен алгоритм p-метода Полларда, включая использование двух указателей (медленного и быстрого) для обнаружения коллизий.

Проделанная работа:

2. **Практическая часть:**
   - Реализована функция для итеративного обновления значений c, u, и v в соответствии с определённым случайным отображением:
     - Если $c < r$: обновляется показатель u, связанный с основанием a.
     - Если $c \geq r$: обновляется показатель v, связанный с основанием b.
   - Реализован поиск коллизий между двумя указателями (медленным и быстрым).
   - Разработана функция для решения уравнения:

$$\Delta v \cdot x \equiv \Delta u \pmod{r},$$

     с использованием вычисления обратного элемента через расширенный алгоритм Евклида.
   - Проведена проверка правильности найденного значения x путём подстановки в исходное уравнение.

3. **Тестирование алгоритма:**
   - Алгоритм протестирован на следующих данных:

$$10^x \equiv 64 \pmod{107},$$

$$a^x \equiv b \pmod{p},$$

$$p = 107, \quad a = 10, \quad r = 53, \quad b = 64.$$

   - На 11-м шаге обнаружена коллизия, и вычислено значение x.

Скриншоты кода

```
p = 107
a = 10
r = 53
b = 64

# Функция, определяющая преобразование
function funf(h, j, k)
    if h < r
        j += 1
        return mod(a * h, p), j, k
    else
        k += 1
        return mod(b * h, p), j, k
    end
end

u, v = 2, 2
U, V = 2, 2

c = mod(a^u * b^v, p)
d = c

println("Начальное значение c: ", c)
println("Начальное значение d: ", d)
```

```
# Применяем преобразование для c и d
c, u, v = funf(c, u, v)
d, U, V = funf(d, U, V)
d, U, V = funf(d, U, V)

println("Обновленное значение c: ", c)
println("Обновленное значение d: ", d)


function second(c, d, u, v, U, V)
    while c != d
        c, u, v = funf(c, u, v)
        d, U, V = funf(d, U, V)
        d, U, V = funf(d, U, V)
        println("Текущее значение c: $c, d: $d")
    end
    return c, d, u, v, U, V
end
```

```
c, d, u, v, U, V = second(c, d, u, v, U, V)

println("Итоговое значение c: ", c)
println("Итоговое значение d: ", d)
println("Итоговое значение u: ", u)
println("Итоговое значение v: ", v)
println("Итоговое значение U: ", U)
println("Итоговое значение V: ", V)
```

Скриншоты кода

```
# Функция для вычисления x из логарифмов
function compute_x(u, v, U, V, r)
    delta_v = mod(v - V, r)
    delta_u = mod(U - u, r)

    if delta_v == 0
        return "Решений нет"
    end

    delta_v_inv = nothing

    try
        delta_v_inv = invmod(delta_v, r)
    catch
        return "Решений нет"
    end

    x = mod(delta_u * delta_v_inv, r)
    return x
end
```

```
function invmod(a, m)
    g, x, _ = gcdx(a, m)
    if g != 1
        throw(ArgumentError("Обратного элемента не существует"))
    else
        return mod(x, m)
    end
end


x = compute_x(u, v, U, V, r)

println("Логарифм x: ", x)
```

Итог

```
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
```

Результаты работы:

- Значение дискретного логарифма:

$$x = 20 \pmod{53}.$$

- Проверка:

$$10^{20} \equiv 64 \pmod{107},$$

что подтверждает корректность алгоритма.

Выводы

1. Алгоритм p-метода Полларда успешно реализован и протестирован. Он позволяет эффективно находить дискретный логарифм в конечных полях.
2. Вычисленные значения u, v, U, V при коллизии подтверждают корректность обновлений логарифмов в процессе работы алгоритма.
3. Реализация функций для работы с конечными полями, включая нахождение обратных элементов, показала высокую точность и стабильность.
4. Основная сложность заключалась в правильной реализации и проверке каждого шага алгоритма, включая корректность работы с модульной арифметикой.
5. Итоговый результат может быть использован для изучения задач дискретного логарифмирования и в дальнейшем применён в криптографических протоколах.

Спасибо за внимание!