Защита лабораторной работы №7

Дискретное логарифмирование в конечном поле

Бурдина К. П.

9 декабря 2023

Российский университет дружбы народов, Москва, Россия

Докладчик

- Бурдина Ксения Павловна
- студентка группы НФИмд-02-23
- студ. билет № 1132236896
- Российский университет дружбы народов
- 1132236896@rudn.ru

Вводная часть

Цель выполнения лабораторной работы

- Освоение дискретного логарифмирования в конечном поле, которое применяется во многих алгоритмах криптографии с открытым ключом
- Программная реализация представленного алгоритма дискретного логарифмирования в конечном поле

Задача дискретного логарифмирования применяется во многих алгоритмах криптографии с открытым ключом.

Пусть $m \in N, m > 1$. Целые числа a и b называются *сравнимыми по модулю m* (обозначается $a \equiv b \, (mod \, m)$), если разность $a - b$ делится на m. Некоторые свойства отношения сравнимости:

- *Рефлексивность*: $a \equiv a \, (mod \, m)$.
- *Симметричность*: если $a \equiv b \, (mod \, m)$, то $b \equiv a \, (mod \, m)$.
- *Транзитивность*: если $a \equiv b \, (mod \, m)$ и $b \equiv c \, (mod \, m)$, то $a \equiv c \, (mod \, m)$.

Отношение, обладающее свойством рефлесивности, симметриности и транзитивности, называется *отношением эквивалентности*. Отношение сравнимости является отношением эквивалентности на множестве Z целых чисел.

Отношение эквивалентности разбивает множество, на котором оно определено, на *классы эквивалентности*. Любые два класса эквивалентности либо не пересекаются, либо совпадают.

Классы эквивалентности, определяемые отношением сравнимости, называются *классами вычетов по модулю m*.

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

Безопастность соответствующих криптосистем основана на том, что решить $a^x\ (mod\,p)$ легко, а задачу дискретного логарифмирования трудно. Поэтому для отображения f чаще всего используются ветвящиеся отображения, например:

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

Алгоритм, реализующий р-метод Полларда

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.

Выход. Показатель x, для которого $a^x \equiv b\,(mod\,p)$, если такой показатель существует.

- выбрать произвольные целые числа u, v и положить $c \leftarrow a^u b^v$ $(mod\,p)$, $d \leftarrow c$
- выполнять $c \leftarrow f(c)(modp)$, $d \leftarrow f(f(d))(modp)$, вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r, до получения равенства $c \equiv d\,(mod\,p)$
- приравняв логарифмы для c и d, вычислить логарифм x решением сравнения по модулю r. Результат: x или “Решений нет”

Решим задачу дискретного логарифмирования $10^x \equiv 64\,(mod\,107)$, используя р-Метод Полларда. Порядок числа 10 по модулю 107 равен 53. Выберем отображение $f(c) = 10c\,(mod\,107)$ при $c < 53$, $f(c) = 64c\,(mod\,107)$ при $c \geq 53$. Пусть $u = 2, v = 2$. Результаты вычислений представлены в таблице:

Номер шага	c	$\log_a c$	d	$\log_a d$
0	4	2+2x	4	2+2x
1	40	3+2x	76	4+2x
2	79	4+2x	56	5+3x
3	27	4+3x	75	5+5x
4	56	5+3x	3	5+7x
5	53	5+4x	86	7+7x
6	75	5+5x	42	8+8x
7	92	5+6x	23	9+9x
8	3	5+7x	53	11+9x
9	30	6+7x	92	11+11x
10	86	7+7x	30	12+12x
11	47	7+8x	47	13+13x

Figure 1: Схема работы алгоритма

Результат выполнения лабораторной работы

Результат выполнения лабораторной работы

Постановка задачи:

- Реализовать задачу дискретного логарифмирования с помощью р-метода Полларда
- Решить задау по заданным числам

Алгоритм Евклида:

```
def alg_e_ext(a, b):
    if b == 0:
        return a, 1, 0
    else:
        d, x, y = alg_e_ext(b, a % b)
    return d, y, x - (a // b) * y
```

```
def inv(a, n):
    return alg_e_ext(a, n)[1]
```

Figure 2: Расширенный алгоритм Евклида

Результат выполнения лабораторной работы

Функция для подсчета значений при выполнении алгоритма Полларда:

```
def fun(x, a, b, xxx):
    (G, H, P, Q) = xxx
    sub = x % 3
    if sub == 0:
        x = x * xxx[0] % xxx[2]
        a = (a + 1) % Q
    if sub == 1:
        x = x * xxx[1] % xxx[2]
        b = (b + 1) % xxx[2]
    if sub == 2:
        x = x * x % xxx[2]
        a = a * 2 % xxx[3]
        b = b * 2 % xxx[3]
    return x, a, b
```

Figure 3: Вспомогательная функция

Алгоритм, реализующий р-метод Полларда:

```
def Pollard(G, H, P):
    Q = int((P - 1) // 2)
    x = G * H
    a = 1
    b = 1
    X = x
    A = a
    B = b
    for i in range(1, P):
        x, a, b = fun(x, a, b, (G, H, P, Q))
        X, A, B = fun(X, A, B, (G, H, P, Q))
        X, A, B = fun(X, A, B, (G, H, P, Q))
        if x == X:
            break
    nom = a - A
    denom = B - b
    res = (inv(denom, Q) * nom) % Q
    if prov(G, H, P, res):
        return res
    return res + Q
```

Figure 4: р-Метод Полларда

Пример реализации алгоритма:

```
def prov(g, h, p, x):
    return pow(g, x, p) == h
args = [(10, 64, 107)]
for arg in args:
    res = Pollard(*arg)
    print(arg, ' : ', res)
    print("Validates: ", prov(arg[0], arg[1], arg[2], res))
```

```
(10, 64, 107)  :  20
Validates:  True
```

Figure 5: Пример реализации

Выводы

Выводы

1. Изучили дискретное логарифмирование в конечном поле
2. Программно реализовали представленный алгоритм дискретного логарифмирования с помощью р-Метода Полларда
3. Решили задачу дискретного логарифмирования для заданных значений