

Отчёт по лабораторной работе №7

Дискретное логарифмирование

Фань Яньцзе

Содержание

1 Цель работы	4
2 Теоретические сведения	5
2.1 р-алгоритм Поллрада	5
3 Выполнение работы	7
3.1 Реализация алгоритма на языке Python	7
3.2 Контрольный пример	10
4 Выводы	11
Список литературы	12

List of Figures

3.1 Работа алгоритма	10
--------------------------------	----

1 Цель работы

Изучение задачи дискретного логарифмирования.

2 Теоретические сведения

Пусть в некоторой конечной мультиликативной абелевой группе G задано уравнение

$$g^x = a$$

Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа x , удовлетворяющего уравнению. Если оно разрешимо, у него должно быть хотя бы одно натуральное решение, не превышающее порядок группы. Это сразу даёт грубую оценку сложности алгоритма поиска решений сверху — алгоритм полного перебора нашёл бы решение за число шагов не выше порядка данной группы.

Чаще всего рассматривается случай, когда группа является циклической, порождённой элементом g . В этом случае уравнение всегда имеет решение. В случае же произвольной группы вопрос о разрешимости задачи дискретного логарифмирования, то есть вопрос о существовании решений уравнения, требует отдельного рассмотрения.

2.1 p -алгоритм Поллрада

- 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 $a^x = b \pmod{p}$, если такой показатель существует.
1. Выбрать произвольные целые числа u, v и положить $c = a^u b^v \pmod{p}$, $d = c$
 2. Выполнять $c=f(c) \pmod{p}$, $d=f(f(d)) \pmod{p}$, вычисляя при этом логарифмы для c и d как линейные функции от x по модулю r , до получения равенства $c = d \pmod{p}$
 3. Приняв логарифмы для c и d , вычислить логарифм x решением сравнения по модулю r . Результат x или РЕШЕНИЯ НЕТ.

3 Выполнение работы

3.1 Реализация алгоритма на языке Python

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

def inverse(a, n):
    return ext_euclid(a, n)[1]

def xab(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

def pollrad(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 = xab(x, a, b, (G, H, P, Q))
        X, A, B = xab(X, A, B, (G, H, P, Q))
        X, A, B = xab(X, A, B, (G, H, P, Q))

    if x == X:
        break

```

```

nom = a-A
denom = B-b
res = (inverse(denom, Q)*nom)%Q

if verify(G, H, P, res):
    return res

return res + Q

def verify(g, h, p, x):
    return pow(g, x, p) == h

args = [(10, 64, 107)]

for arg in args:
    res = pollrad(*arg)
    print(arg, " : ", res)
    print("Validates: ", verify(arg[0], arg[1], arg[2], res))

```

3.2 Контрольный пример

```
55     return res
56
57     return res + Q
58
59 def verify(g, h, p, x):
60     return pow(g, x, p) == h
61
62 args = [(10, 64, 107)]
63
64 for arg in args:
65     res = pollrad(*arg)
66     print(arg, ' : ', res)
67     print("Validates: ", verify(arg[0], arg[1], arg[2], res))
68
(10, 64, 107) : 20
Validates: True
```

[1]: 1

Figure 3.1: Работа алгоритма

4 Выводы

Изучили задачу дискретного логарифмирования.

Список литературы

1. Дискретное логарифмирование)
2. Доступно о криптографии на эллиптических кривых