Дискретное логарифмирование в конечном поле

Каримов Зуфар Исматович

2022 Moscow, Russia

RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

Реализация алгоритма, реализующий p-метод Полларда для задач дискретного логарифмирования.

Задачи

Задачи

1. Реализовать алгоритм, реализующий p-метод Полларда для задач дискретного логарифмирования.

Реализация

1. Написал функцию ext_euclid и inverse (рис. 1)

Ввод [7]:
```
def ext_euclid(a, b):
    """
    Extended Euclidean Algorithm
    :param a:
    :param b:
    :return:
    """
    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
```

Ввод [8]:
```
def inverse(a, n):
    """
    Inverse of a in mod n
    :param a:
    :param n:
    :return:
    """
    return ext_euclid(a, n)[1]
```

Figure 1: Функция для расширенного алгоритма Евклида и обратного значнения

2. Написал функцию xab (рис. 2)

Ввод [9]:

```
def xab(x, a, b, xxx_todo_changeme):
    """
    Pollard Step
    :param x:
    :param a:
    :param b:
    :return:
    """
    (G, H, P, Q) = xxx_todo_changeme
    sub = x % 3 # Subsets

    if sub == 0:
        x = x*xxx_todo_changeme[0] % xxx_todo_changeme[2]
        a = (a+1) % Q

    if sub == 1:
        x = x * xxx_todo_changeme[1] % xxx_todo_changeme[2]
        b = (b + 1) % xxx_todo_changeme[2]

    if sub == 2:
        x = x*x % xxx_todo_changeme[2]
        a = a*2 % xxx_todo_changeme[3]
        b = b*2 % xxx_todo_changeme[3]

    return x, a, b
```

Figure 2: Функция xab

3. Написал функцию pollard (рис. 3)

Ввод [10]:

```python
def pollard(G, H, P):
    # P: prime
    # H:
    # G: generator
    Q = int((P - 1) // 2)  # sub group
    x = G*H
    a = 1
    b = 1
    X = x
    A = a
    B = b

    # Do not use range() here. It makes the algorithm amazingly slow.
    for i in range(1, P):
        # Who needs pass-by reference when you have Python!!! ;)
        # Hedgehog
        x, a, b = xab(x, a, b, (G, H, P, Q))
        # Rabbit
        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
    # print nom, denom
    # It is necessary to compute the inverse to properly compute the fraction mod q
    res = (inverse(denom, Q) * nom) % Q
    # так никто не делает но все же...
    if verify(G, H, P, res):
        return res

    return res + Q
```

Figure 3: Функция для алгоритма pollard

4. Написал функцию verify и блок работы программы(рис. 4)

Ввод [11]:
```
def verify(g, h, p, x):
    """
    Verifies a given set of g, h, p and x
    :param g: Generator
    :param h:
    :param p: Prime
    :param x: Computed X
    :return:
    """
    return pow(g, x, p) == h

args = [
    (10, 64, 107),
]
```

Ввод [12]:
```
for arg in args:
    res = pollard(*arg)
    print(arg, ': ', res)
    print("Validates: ", verify(arg[0], arg[1], arg[2], res))
    print()
```

Figure 4: Функция verify и блок работы программы

Результат

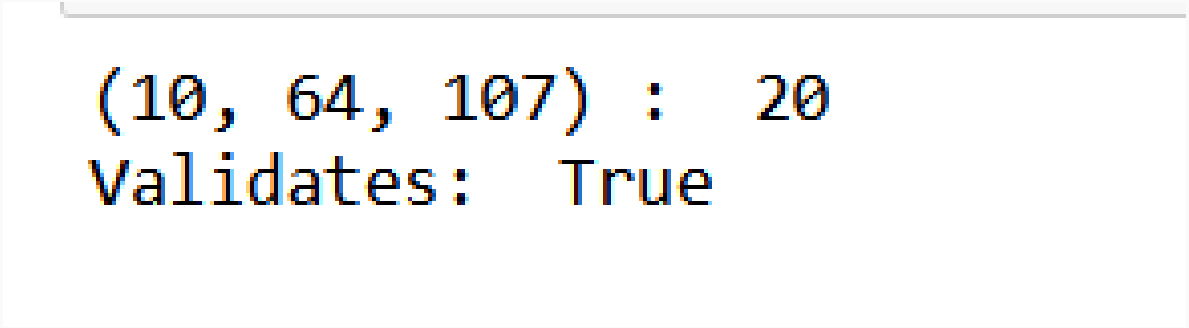

Figure 5: Результат алгоритма

Вывод

Реализовал реализующий p-метод Полларда для задач дискретного логарифмирования.

Спасибо за внимание