Лабораторная работа №7

Презентация

Миличевич Александра

15 февраля 2025

Российский университет дружбы народов, Москва, Россия

Цель лабораторной работы

Цель лабораторной работы №7 заключается в ознакомлении студентов с алгоритмом Полларда для дискретного логарифмирования. Студенты должны изучить и реализовать алгоритм, который позволяет находить дискретные логарифмы в конечных полях, а также понять его применение в криптографии и теории чисел.

P-алгоритм Полларда — это метод для вычисления дискретного логарифма в конечных полях, который основан на идее разбиения задачи на более простые подзадачи. Алгоритм использует метод “разделяй и властвуй”, комбинируя два подхода: метод грубой силы и метод “смешивания” (baby-step giant-step). Он строит таблицу значений для малых степеней и использует их для нахождения соответствующих значений для больших степеней, что значительно ускоряет процесс. P-алгоритм Полларда эффективен для больших чисел и широко применяется в криптографии, особенно в системах, основанных на эллиптических кривых.

1. Функция `modular_inverse(a, n)`

Эта функция вычисляет обратное к `a` по модулю `n`.

Описание:

Вход:

- `a` (int): Число, для которого ищется обратное.
- `n` (int): Модуль.

Выход:

Обратное к `a` по модулю `n`.

Как работает:

1. Используется функция `extended_euclidean(a, n)` для получения коэффициентов Безу.
2. Возвращается коэффициент `x` (второй элемент в кортеже), который является обратным к `a` по модулю `n`.
3. Возвращается коэффициент `x` (второй элемент в кортеже), который является обратным к `a` по модулю `n`.

```python
def modular_inverse(a, n):
    """
    Вычисляет обратное к 'a' по модулю 'n'.

    Args:
        a (int): Число, для которого ищется обратное.
        n (int): Модуль.

    Returns:
        int: Обратное к 'a' по модулю 'n'.
    """
    return extended_euclidean(a, n)[1]
```

Рис. 1: modular inverse

2. Функция `pollard_step(x, a, b, params)`

Эта функция реализует один шаг алгоритма Полларда для дискретного логарифмирования.

Описание:

Вход:

- `x` (int): Текущее значение `x`.
- `a` (int): Текущее значение `a`.
- `b` (int): Текущее значение `b`.
- `params` (tuple): Параметры (G, H, P, Q).

Выход:

Кортеж обновленных значений `(x, a, b)`.

Как работает:

Разделение на подмножества:

Использует `x % 3` для определения подмножества.

Обновление значений в зависимости от подмножества:

- Если `x % 3 == 0`: `x` умножается на `G` по модулю `P`, `a` увеличивается на 1 по модулю `Q`.
- Если `x % 3 == 1`: `x` умножается на `H` по модулю `P`, `b` увеличивается на 1 по модулю `Q`.
- Если `x % 3 == 2`: `x` возводится в квадрат по модулю `P`, `a` и `b` умножаются на 2 по модулю `Q`.

```python
def pollard_step(x, a, b, params):
    """
    Шаг алгоритма Полларда для дискретного логарифмирования.

    Args:
        x (int): Текущее значение x.
        a (int): Текущее значение a.
        b (int): Текущее значение b.
        params (tuple): Параметры (G, H, P, Q).

    Returns:
        tuple: Обновленные значения x, a, b.
    """
    G, H, P, Q = params
    subset = x % 3 # Выбираем подмножество

    if subset == 0:
        x = (x * G) % P
        a = (a + 1) % Q

    if subset == 1:
        x = (x * H) % P
        b = (b + 1) % Q

    if subset == 2:
        x = (x * x) % P
        a = (a * 2) % Q
        b = (b * 2) % Q

    return x, a, b
```

Рис. 2: pollard step

3. Функция `pollard_rho_discrete_log(generator, value, prime)`

Эта функция реализует алгоритм Полларда для дискретного логарифмирования.

Описание:

Вход:

- `generator` (int): Генератор группы.
- `value` (int): Значение, для которого ищется дискретный логарифм.
- `prime` (int): Простое число (порядок группы).

Выход:

Дискретный логарифм (если найден) или сообщение об ошибке.

Как работает:

Инициализация:

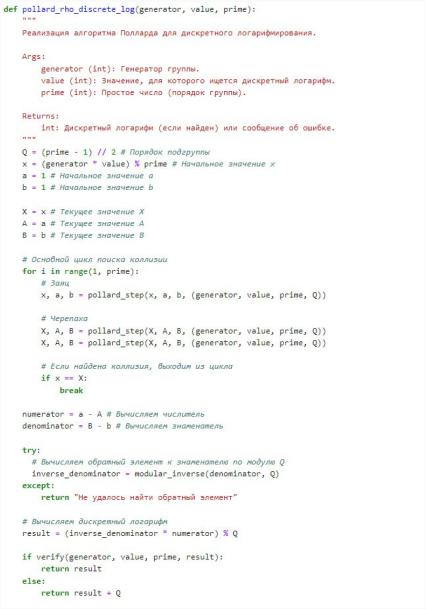
Устанавливаются начальные значения `Q`, `x`, `a`, `b`, `X`, `A`, `B`.

Основной цикл:

- Используются “заяц” и “черепаха” для поиска коллизии, где заяц делает два шага за итерацию, а черепаха - один.
- Функция `pollard_step` применяется для каждого шага.
- Цикл выполняется до тех пор, пока не будет найдена коллизия (`x == X`).

Вычисление дискретного логарифма:

- Вычисляется числитель `a - A` и знаменатель `B - b`.
- Вычисляется обратный элемент знаменателя по модулю `Q` с помощью функции `modular_inverse`.
- Вычисляется дискретный логарифм: `(inverse_denominator * numerator) %`



```python
def pollard_rho_discrete_log(generator, value, prime):
    """
    Реализация алгоритма Полларда для дискретного логарифмирования.

    Args:
        generator (int): Генератор группы.
        value (int): Значение, для которого ищется дискретный логарифм.
        prime (int): Простое число (порядок группы).

    Returns:
        int: Дискретный логарифм (если найден) или сообщение об ошибке.
    """
    Q = (prime - 1) // 2 # Порядок подгруппы
    x = (generator * value) % prime # Начальное значение x
    a = 1 # Начальное значение a
    b = 1 # Начальное значение b

    X = x # Текущее значение X
    A = a # Текущее значение A
    B = b # Текущее значение B

    # Основной цикл поиска коллизии
    for i in range(1, prime):
        # Заяц
        x, a, b = pollard_step(x, a, b, (generator, value, prime, Q))

        # Черепаха
        X, A, B = pollard_step(X, A, B, (generator, value, prime, Q))
        X, A, B = pollard_step(X, A, B, (generator, value, prime, Q))

        # Если найдена коллизия, выходим из цикла
        if x == X:
            break

    numerator = a - A # Вычисляем числитель
    denominator = B - b # Вычисляем знаменатель

    try:
      # Вычисляем обратный элемент к знаменателю по модулю Q
      inverse_denominator = modular_inverse(denominator, Q)
    except:
        return "Не удалось найти обратный элемент"

    # Вычисляем дискретный логарифм
    result = (inverse_denominator * numerator) % Q

    if verify(generator, value, prime, result):
        return result
    else:
        return result + Q
```

Рис. 3: pollard descrete log

Функция `verify(generator, value, prime, x)`

Эта функция проверяет правильность вычисленного дискретного логарифма.

Описание:

Вход:

- `generator` (int): Генератор группы.
- `value` (int): Значение, для которого ищется дискретный логарифм.
- `prime` (int): Простое число (порядок группы).
- `x` (int): Вычисленный дискретный логарифм.

Выход:

`True`, если логарифм верный, `False` в противном случае.

```python
def verify(generator, value, prime, x):
    """
    Проверяет правильность вычисленного дискретного логарифма.

    Args:
        generator (int): Генератор группы.
        value (int): Значение, для которого ищется дискретный логарифм.
        prime (int): Простое число (порядок группы).
        x (int): Вычисленный дискретный логарифм.

    Returns:
        bool: True, если логарифм верный, False в противном случае.
    """
    return pow(generator, x, prime) == value
```

Рис. 4: verify

Вывод

Изучили задачу дискретного логарифмирования.