Оглавление

[1 Теоретическая часть 2](#_Toc152436588)

[1.1.1 Метод Гельфонда-Шенкса 2](#_Toc152436589)

[1.1.2 Алгоритм Метода Гельфонда-Шенкса 2](#_Toc152436590)

[1.1.3 Псевдокод Метода Гельфонда-Шенкса 3](#_Toc152436591)

[1.2.1 метод Полларда 3](#_Toc152436592)

[1.2.2 Алгоритм метода Полларда 4](#_Toc152436593)

[1.2.3 Псевдокод метода Полларда 5](#_Toc152436594)

[1.3.1 Метод вычисления дискретного логарифмирования в конечных полях 6](#_Toc152436595)

[1.3.2 Алгоритм Метода вычисления дискретного логарифмирования в конечных полях 6](#_Toc152436596)

[1.3.3 Псевдокод Метода вычисления дискретного логарифмирования в конечных полях 8](#_Toc152436597)

1 Теоретическая часть

Пусть – конечная циклическая группа порядка , – образующий элемент и . Дискретным логарифмом элемента группы по основанию называется число , являющееся решением уравнения .

Задача дискретного логарифмирования является фундаментальной для анализа целого ряда криптографических протоколов. Для криптографической практики наиболее важными являются следующие циклические группы и проблема дискретного логарифмирования в них:

мультипликативная группа конечного простого поля из элементов;

мультипликативная группа конечного поля из элементов;

циклическая подгруппа порядка группы точек эллиптической кривой над конечным полем .

1.1.1 Метод Гельфонда-Шенкса

Данный алгоритм исторически является одним из первых методов дискретного логарифмирования. Достоинством данного алгоритма является его детерминированный характер, а также отсутствие необходимости знать точное значение порядка группы . Его недостатком является большое время работы.

1.1.2 Алгоритм Метода Гельфонда-Шенкса:

Вход. Конечная циклическая группа , верхняя оценка для порядка группы , элемент .

Выход. Число .

Шаг 1. Вычислить и вычислить элементы , упорядочить массив пар по второй координате.

Шаг 2. Вычислить , для каждого проверить, является ли элемент второй координатой какой-либо пары из упорядоченного на первом шаге массива пар. Если , то запомнить число .

Шаг 3. Среди всех чисел, найденных на втором этапе, выбрать наименьшее, это значение и будет искомым значением .

Сложность метода Гельфонда-Шенкса равна .

1.1.3 Псевдокод Метода Гельфонда-Шенкса:

Процедура Гельфонд\_Шенкс():

*r = взять\_целое(sqrt(b)) + 1*

*ga = создать\_пустой\_список*

*Для каждого a в диапазоне от 0 до r:*

*ga.добавить([a, pow(g, a, b)])*

*ga.сортировать\_по\_ключу(ключ=lambda el: el[1])*

*g1 = pow(mulinv(g, b), r, b)*

*x = b + 1*

*Для каждого i в диапазоне от 0 до r:*

*gh = (pow(g1, i, b) \* h) % b*

*Для каждого элемента в ga:*

*Если элемент[1] == gh:*

*x = минимум(x, элемент[0] + r \* i)*

*Иначе Если элемент[1] > gh:*

*Прервать*

*Конец Для*

*Конец Для*

*Вернуть x*

*Конец Процедуры*

1.2.1 метод Полларда

Дана конечная циклическая группа и известен ее порядок и . Применим метод Полларда к любой циклической группе , чьи элементы представлены таким образом, что их можно разбить на три примерно равные, попарно непересекающиеся части . При этом должен существовать эффективный способ проверки, к какому из этих подмножеств принадлежит данный элемент группы.

Определим функцию на таким образом, что:

Будет построена рекуррентная последовательность . Из определения функции нетрудно заметить, что при любом , для некоторых . Также нетрудно заметить, что последовательности задаются следующими рекуррентными соотношениями:

При вычислении очередного члена последовательности , числа вычисляются по известным очень легко. При этом для любого выполняется равенство .

1.2.2 Алгоритм метода Полларда

Вход. Конечная циклическая группа , порядка , элемент . Выход. Число .

Шаг 1. Вычислить .

Шаг 2. Положить , выбрать случайное , вычислить . Запомнить две тройки и перейти к шагу 4.

Шаг 3. Положить , найти Запомнить две тройки и перейти к шагу 4.

Шаг 4. Если , то проверить выполнение условия . Если это условие выполнено, то перейти к шагу 3. В противном случае остановить алгоритм и сообщить, что вычислить не удалось. Если , то перейти к шагу 5.

Шаг 5. Вычислить . Если , то перейти на шаг 2 и выбрать новое значение . В противном случае решить сравнение . Если то единственное решение сравнения равно искомому . Если же , то сравнение имеет различных решений по модулю . Для каждого из этих решений проверить выполнимость равенства и найти истинное решение .

Сложность -метода Полларда равна .

1.2.3 Псевдокод метода Полларда

Процедура Метод\_Полларда(g, m, h, epsilon):

t = взять\_целое(sqrt(2 \* m \* log(1 / epsilon))) + 1

Пока Истина:

i = 1

s = взять\_случайное\_число\_в\_интервале(0, m - 2)

y0, a0, b0 = next\_elements(pow(g, s, m), s, 0, h, g, m)

y1, a1, b1 = next\_elements(y0, a0, b0, h, g, m)

Пока Истина:

y0, a0, b0 = next\_elements(y0, a0, b0, h, g, m)

y1, a1, b1 = next\_elements(next\_elements(y1, a1, b1, h, g, m), h, g, m)

Если y0 != y1 Тогда:

Если i >= t Тогда

Вернуть Пусто

ИначеЕсли

i += 1

Конец Если

ИначеЕсли

new\_b = (b0 - b1) % (m - 1)

new\_a = (a1 - a0) % (m - 1)

d = НОД(new\_a, m - 1)

Если d > взять\_целое(sqrt(m - 1)) Тогда

Прервать

Конец Если

Для каждого x в решить\_неравенство(new\_a, new\_b, m - 1):

Если pow(g, x, m) == h Тогда

Вернуть x

Конец Для

Прервать

Конец Если

Конец Пока

Конец Пока

Конец Процедуры

1.3.1 Метод вычисления дискретного логарифмирования в конечных полях

Даны – образующий элемент группы и . Требуется найти Будем считать, что .

Пусть – некоторое натуральное число, параметр метода. Определим факторную базу – множество первых простых чисел, не превосходящих , . Значение параметра выбирается таким образом, чтобы минимизировать сложность алгоритма.

Для всех обозначим , где .

Алгоритм аналогичен субэкспоненциальным алгоритмам факторизации, но соответствующие алгоритмы дискретного логарифимирования обладают одной особенностью – эти алгоритмы можно условно разделить на два этапа:

По заданному надо выбрать факторную базу и определить логарифмы элементов факторной базы (при фиксированном этот этап надо проделать только один раз);

С использованием известных логарифмов элементов факторной базы по заданному необходимо найти (этот этап может производиться неоднократно).

1.3.2 Алгоритм Метода вычисления дискретного логарифмирования в конечных полях:

Вход: простое нечетное число , .

Выход: значение .

Шаг 1. Выберем значение параметра . Построить факторную базу .

Шаг 2. Выберем случайное , найти вычет .

Шаг 3. Проверим число на -гладкость. Если является -гладким, то вычислим его каноническое разложение

Из соотношений вытекает сравнение где .

Повторим шаги 2 и 3 до тех пор, пока число найденных строк не превысит , где – некоторая небольшая константа. В результате будет построена система линейных уравнений над кольцом относительно известных

Полученная система заведомо совместна.

Шаг 4. Решим полученную на предыдущем шаге систему линейных уравнений над кольцом . Если система имеет более одного решения, то вернемся к шагу 2 и получим несколько новых линейных соотношений. Затем вернемся к шагу 4.

Шаг 5. (Вычисление индивидуального логарифма.) Выберем случайное , найдем вычет . Проверим число на -гладкость. Если является -гладким, то и, следовательно,

При оптимальное значение и сложность всего алгоритма оценивается величиной , где:

для

1.3.3 Псевдокод Метода вычисления дискретного логарифмирования в конечных полях:

Процедура Дискретный\_логарифм(g, h, p):

m = порядок\_группы\_поля(p) # Порядок конечного поля

t = взять\_целое(sqrt(m)) + 1

Для каждого i в диапазоне от 0 до t:

y = pow(g, i \* t, p)

inv\_y = обратный\_элемент(y, p) # Обратный элемент к y в конечном поле

u = (h \* inv\_y) % p

# Решение дискретного логарифма в подгруппе порядка t

x = solve\_discrete\_logarithm(g, u, p, t)

Если x не равно Пусто:

# Найдено значение дискретного логарифма

Вернуть i \* t + x

Конец Для

# Если ни одно значение не было найдено

Вернуть Пусто

Конец Процедуры

Процедура solve\_discrete\_logarithm(g, u, p, t):

# Решение дискретного логарифма в подгруппе порядка t

Для каждого j в диапазоне от 0 до t:

Если pow(g, j, p) == u:

Вернуть j

Конец Для

# Если решение не найдено

Вернуть Пусто

Конец Процедуры