Алгоритмы дискретного логарифмирования

1. Постановка задачи
2. Метод «шаг младенца, шаг великана»
3. Алгоритм исчисления порядка
4. Задачи и упражнения

1. Постановка задачи

Для построения надежной криптосистемы необходимо принимать во внимание те методы взлома, которые может применить злоумышленник, и выбирать параметры криптосистемы (в частности, длины чисел) так, чтобы сделать эти методы практически нереализуемыми. Рассмотрим два таких метода, для того чтобы дать некоторое представление об этой «таинственной» области.

Долгое время эффективный алгоритм решения задачи дискретного логарифмирования не был известен и при вычислениях использовались заранее подготовленные таблицы индексов. В 1962 году советским математиком Александром Осиповичем Гельфандом был предложен метод, который позволил вычислять индексы достаточно эффективно при небольших значениях p. В русскоязычной литературе этот метод получил название «метод согласования». Это был один из первых методов, более быстрый чем метод прямого перебора.

Общая схема алгоритма такова: Берем два целых числа m и k, таких что mk>p (как правило, m=k=$\lceil\sqrt{p}\rceil + 1$). Затем вычисляются два ряда чисел:

$$a, ga, g^2a, \dots, g^{m-1}a \ (mod\ p)$$
$$g^m, g^{2m}, g^{3m}, \dots, g^{km} \ (mod\ p)$$

(все вычисления произведены по модулю p). Найдем такие i и j, для которых $g^i a = g^{jm}$. Тогда x = jm – i. Справедливость последнего равенства подтверждается следующей цепочкой, все вычисления в которой произведены по модулю p:

$$g^x = g^{jm-i} = g^{jm} \cdot \left(g^i\right)^{-1} = g^{jm}a \cdot \left(g^i a\right)^{-1} = g^{jm}a \cdot \left(g^{jm}\right)^{-1} = a$$

Заметим, что числа i и j непременно будут найдены, поскольку при i=0,…, m –1, j=1,…, k выполняется jm–i=1,…,km , причем km>p. То есть среди всех чисел вида jm–i обязательно содержится $0 < x \le p$.

Замечание: Указанный метод можно применять для разыскания дискретных логарифмов в любой циклической группе порядка n. Приведем этот метод в форме алгоритма.

Алгоритм «Шаг младенца-шаг великана»:

Вход: g - порождающий элемент конечной группы G порядка n; a ∈ G.

Шаг 1. Вычислить m=k=$\lceil\sqrt{p}\rceil + 1$.

Шаг 2. Вычислить $b = g^m$.

Шаг 3. Вычислить последовательности $u_i = b^i,\ v_j = ag^j$. Для i, j = 1,…, m.

Шаг 4. Найти i, j такие что $u_i = v_j => x = mi - j\ mod\ n$.

Вывод: $\log_g a = x$.

Одна из трудоемких частей этого алгоритма – это поиск на Шаге 4. Он может быть осуществлен несколькими способами:

1) Сначала построить таблицу (i, u_i), отсортировать ее по второй компоненте а затем производить сравнения по мере нахождения компонент v_j.

2) Построить две таблицы (i, u_i) и (j, v_j), отсортировать каждую из них, а затем произвести поиск совпадений.

3) Объединить u, v в одну таблицу, снабдив их номером в соответствующей последовательности и битом принадлежности к одной из двух последовательностей, а затем применить совместную сортировку.

Сложность данного алгоритма составляет $O(\sqrt{n})$ умножений по модулю и $O(\sqrt{n}\ln n)$ операций сравнения.

Пример. Пусть n=229 (простое число), g=6, a=12.

Шаг 1. $m = \lceil\sqrt{229}\rceil + 1 = 15 + 1 = 16.$

Шаг 2. $b = g^m \ mod\ n = 6^{16}\ mod\ 229 = 183$

Шаг 3. В этом примере вычислим сначала ряд $u_i = b^i = 183^i$, а затем будем вычислять компоненты $v_j = ag^j = 12 \cdot 6^j$ до тех пор, пока не найдется совпадение.

i, j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
u_i	183	55	218	48	82	121	159	14	43	83	75	214	3	91	165	**196**
v_j	72	203	73	209	109	**196**										

i=16, j=6. x=mi – j mod n = 250 mod 228 = 22.

Проверка: $6^{22}\ mod\ 229 = 12.$

Ответ: $\log_6 12\ \ mod\ 228\ \ =\ \ 22.$

Алгоритм исчисления порядка (index-calculus algorithm)

Основные идеи алгоритма исчисления порядка были известны с 20-х годов XX века, но лишь в 1979 году Адлерман указал на этот алгоритм как на средство вычисления дискретного логарифма и исследовал его трудоемкость. В настоящее время алгоритм исчисления порядка и его улучшенные варианты дают наиболее быстрый способ вычисления дискретных логарифмов в некоторых конечных группах, в частности, в группе Z_p^* . Этот алгоритм в отличие от алгоритмов прямого поиска и ро-метода подходит не для всех циклических групп.

Алгоритм исчисления порядка

Вход: g – порождающий элемент циклической группы порядка n, a ∈ G, c≈10 – параметр надежности.

Вход: g – порождающий элемент циклической группы порядка n, a∈G, c≈10 – параметр надежности.

Ш.1. Выбирается факторная база $S=\{p_1, p_2, \ldots, p_t\}$. (Если $G=Z_p^*$, то S состоит из t первых простых чисел.)

Ш.2. Выбрать случайное k: 0≤k<n и вычислить g^k.

Ш.3. Попытаться разложить g^k по факторной базе:

$$g^k = \prod_{i=1}^{t} p_i^{\alpha_i},\ \alpha_i \geq 0.$$

Если это не удалось, вернуться на Шаг 2.

Ш.4. Логарифмируя обе части получившегося выражения, получаем

$$k \equiv \sum_{i=1}^{t} \alpha_i \log_g p_i \ (mod\ n) \quad *$$

В этом выражении неизвестными являются логарифмы.

Это сравнение с t неизвестными следует запомнить.

Ш.5. Если сравнений вида (*), полученных на Шаге 4, меньше, чем t+c, то вернуться на Шаг 2.

Ш.6. Решить систему t+c сравнений с t неизвестными вида (*), составленную на Шагах 2-5.

Ш.7. Выбрать случайное k: 0≤k<n и вычислить ag^k.

Ш.8. Попытаться разложить ag^k *по факторной базе:*

$$ag^k = \prod_{i=1}^{t} p_i^{\beta_i}, \ \beta_i \geq 0.$$

Если это не удалось, вернуться на Шаг 7.

Ш.9. Логарифмируя обе части последнего равенства, получаем

$$x = \log_g a = \sum_{i=1}^{t} \beta_i \log_g p_i - k \bmod n,$$

где $\log_g p_i$ *(1≤i≤t) вычислены на Шаге 6 как решение системы сравнений.*

Выход:

$x = \log_g a \bmod n$.

В том случае, когда $G=Z_p^*$, в качестве факторной базы S берут *t* первых простых чисел. Такой выбор оправдан следующим наблюдением. Число, наугад выбранное из множества целых чисел, с вероятностью 1/2 делится на 2, с вероятностью 1/3 – на 3, с вероятностью 1/5 – на 5 и т.д. Поэтому можно ожидать, что в промежутке от 1 до *p*—1 найдется достаточно много чисел, в разложении которых участвуют только маленькие простые делители из множества S. Именно такие числа отыскиваются на шагах 2 и 7.

Параметр *c* вводится для того, чтобы система сравнений, решаемая на Шаге 6, имела единственное решение. Дело в том, что полученная система может содержать линейно зависимые сравнения. Считается, что при значении *c* порядка 10 и большом *p* система сравнений имеет единственное решение с высокой вероятностью.

Пример:
$G = \mathbb{Z}_{71}^*, g = 7, a = 26, \ n = \varphi(71) = 70.$
S={2, 3, 5, 7} (Шаг 1). (Можем сразу указать $\log_7 7 \ mod \ 70 = 1$).
Теперь будем перебирать *k* для составления системы уравнений вида * (Шаги 2-5).
$k = 2, => 7^2 \ mod \ 71 = 49 = 7 \cdot 7$ (поскольку $\log_7 7$ уже вычислен, это сравнение нам не пригодится).

k=3, 7^3 mod 71=59.

k=4, 7^4 mod 71=58=2·29.

k=5, 7^5 mod 71=51=3·17.

k=6, 7^6 mod 71=2 $\Rightarrow$ $6\equiv\log_7 2 \pmod{70}$

k=7, 7^7 mod 71=14=2·7 $\Rightarrow$ $7\equiv\log_7 2+\log_7 7 \pmod{70}$

k=8, 7^8 mod 71=27=3^3 $\Rightarrow$ $8\equiv 3\log_7 3 \pmod{70}$

k=9, 7^9 mod 71=47.

k=10, 7^{10} mod 71=45=3^2·5 $\Rightarrow$ $10\equiv 2\log_7 3+\log_7 5 \pmod{70}$

Теперь имеем достаточно сравнений для того, чтобы определить логарифмы от элементов факторной базы. Вот эти сравнения:

$6\equiv\log_7 2 \pmod{70}$

$7\equiv\log_7 2+\log_7 7 \pmod{70}$

$8\equiv 3\log_7 3 \pmod{70}$

$10\equiv 2\log_7 3+\log_7 5 \pmod{70}$

Решая полученную систему, получаем (Шаг 6):

$\log_7 2\equiv 6 \pmod{70}$, $\log_7 3\equiv 26 \pmod{70}$,

$\log_7 5\equiv 28 \pmod{70}$, $\log_7 7\equiv 1 \pmod{70}$.

Перейдем к Шагам 7—9:

k=1, 26·7 mod 71=40=2^3·5 $\Rightarrow$ $\log_7 26\equiv 3\log_7 2+\log_7 5—1 \pmod{70}$ $\Rightarrow$

$\log_7 26\equiv 3\cdot 6+28—1 \pmod{70}$

$\log_7 26\equiv 45 \pmod{70}$

Проверка:

7^{45} mod 71 = 26. Верно.

Ответ:

$\log_7 26\equiv 45 \pmod{70}$.

Замечание:

Для случая G=Z_p и для случая G=F_{2^m} составляет $L_q[1/2,c]$, где q есть мощность G, $c > 0$ – константа.

Решите задачу дискретного логарифмирования

$$2^x \equiv 103 \pmod{443}, \quad 2^x \equiv 120 \pmod{467},$$
$$3^x \equiv 267 \pmod{487}, \quad 3^x \equiv 118 \pmod{521}.$$

Используя метод Ферма, разложите на множители числа 197881, 295543 и 327653.