

# ОБЧИСЛЕННЯ ДИСКРЕТНОГО ЛОГАРИФМУ В ХМАРНИХ СИСТЕМАХ

Б. С. Рибак<sup>1, а</sup>

<sup>1</sup>Національний технічний університет України «Київський політехнічний інститут»

## Анотація

У цій роботі аналізуються підходи до побудови швидкого паралельного алгоритму дискретного логарифмування для застосування у системах хмарних обчислень. Також проводиться огляд цих систем і обирається найгнучкіша з огляду на можливість виділення необхідної кількості ресурсів за прийнятною ціною.

**Ключові слова:** дискретний логарифм, index-calculus, хмарні обчислення

## Вступ

Стійкість великої кількості асиметричних крипто-систем, що існують у світі, ґрунтується на складності розв'язання проблеми дискретного логарифма. Найбільш відомими прикладами криптографічних протоколів, що використовують цю складність для забезпечення захисту інформації, є схеми шифрування та електронного підпису Ель-Гамала, схема обміну ключами Діффі-Хеллмана тощо. Для варіацій проблеми дискретного логарифму, що використовуються у цих протоколах, зокрема, логарифму в групах точок еліптичних кривих та в мультиплікативних групах простих полів  $Z_p^*$  на даний момент не є відомими алгоритми, що мають поліноміальну складність у часі.

Наразі у зв'язку з поширенням багатоядерних процесорів та розподілених обчислень необхідно враховувати можливість паралелізації виконання алгоритму як в рамках одного обчислювального вузла, так і у рамках декількох, з'єднаних локальною мережею. Популярною є модель забезпечення зручного доступу через мережу до спільного пулу обчислювальних ресурсів, що можуть бути оперативно надані та вивільнені з мінімальними управлінськими затратами та зверненнями до провайдера — модель так званих *хмарних обчислень* [1].

Метою роботи є визначення доцільного алгоритму для побудови процесу обчислення дискретного логарифму у окремих алгебраїчних групах з урахуванням необхідності паралельних обрахунків та огляд існуючих платформ хмарних обчислень з визначенням найоптимальнішої для задачі.

## 1. Обчислення дискретного логарифму

### 1.1. Постановка задачі

Наведемо загальне визначення задачі згідно до [2]. Нехай задана деяка група  $G$ . Для елемента  $g \in G$  позначимо згенеровану цим елементом циклічну під-

групу як  $\langle g \rangle$ . Ціле число  $x$  називається *дискретним логарифмом* елемента  $a$  за основою  $g$  у групі  $G$ , якщо  $g^x = a$  для заданих  $g \in G$  та  $a \in \langle g \rangle$

Існують декілька типів груп, що є найчастіше вживаними при розробці криптосистем. До них відносяться групи  $Z_p^*$ , де  $p$  — просте, мультиплікативні групи  $Z_{2^n}^*$ ,  $n \in \mathbb{N}$  та підгрупи точок еліптичної кривої над скінченним полем.

### 1.2. Алгоритми дискретного логарифмування

Для простіших позначень у подальшому описі алгоритмів вважається, що ми працюємо у групі  $Z_p^*$  порядку  $p - 1$ , де  $p$  — просте, якщо не зазначено інше.

#### 1.2.1. Прості алгоритми

Найпростішим алгоритмом розв'язання задачі є, очевидно, повний перебір всіх можливих значень  $x$  з перевіркою  $g^x$  і співставленні результату з  $a$  на кожному кроці. Цей алгоритм вимагає  $O(p)$  операцій у гіршому випадку.

Більш швидким є наступний алгоритм, запропонований Шенксом [3], що використовує один з традиційних підходів до побудови обчислювальних процедур — Meet in the Middle (зустріч посередині). Інша назва цього алгоритму, що походить від його структури — Baby-Step-Giant-Step. Він є визначеним, наприклад, у [4]. Наведемо його у формі словесного опису:

- 1) Присвоїти  $H := \lfloor p^{1/2} \rfloor + 1$ .
- 2) Знайти  $c \equiv a^H \pmod{p}$ .
- 3) Скласти таблицю значень  $c^u \pmod{p}$ ,  $1 \leq u \leq H$  та впорядкувати.
- 4) Скласти таблицю значень  $b \cdot a^v \pmod{p}$ ,  $0 \leq v \leq$  та впорядкувати.
- 5) Знайти однакові елементи з першої та другої таблиць. Для них:

$$c^u \equiv b \cdot a^v \pmod{p}.$$

Звідки маємо  $a^{Hu-v} \equiv b \pmod{p}$ .

<sup>а</sup>bohdan.rybak@gmail.com

6) Вивести відповідь, як  $x \equiv Hu - v \pmod{p}$ .

Алгоритм Шенкса має складність у  $O(p^{1/2} \log p)$  операцій, але потребує великої кількості пам'яті для зберігання проміжних таблиць.

### 1.2.2. Алгоритм Поліга-Хеллмана

Припустимо, що нам є відомим розклад числа  $p - 1$  на прості множники:

$$p - 1 = \prod_{i=1}^s q_i^{\alpha_i}. \quad (1)$$

Тоді існують алгоритми, що використовують цю інформацію для пошуку дискретного логарифму. Одним з найвідоміших є алгоритм Поліга-Хеллмана, словесний опис якого згідно до [4, с.130] наводиться далі.

Нехай розклад (1) є відомим.

*Крок 1.* Для кожного простого числа  $q$ ,  $q | p - 1$  та для кожного цілого  $j = 0, \dots, q - 1$  створюємо таблицю чисел

$$r_{qj} = g^{\frac{j(p-1)}{q}} \pmod{p}.$$

*Крок 2.* Для кожного простого дільника  $q_i$  у розкладі  $p - 1$  знаходимо  $\log_g a \pmod{q_i^{\alpha_i}}$ .

Детальніше опишемо механізм його знаходження. Нехай  $x \equiv \log_g a \pmod{q_i^{\alpha_i}}$ . Представляємо його у вигляді:

$$x \equiv x_0 + x_1 q + \dots + x_{\alpha_i - 1} q^{\alpha_i - 1} \pmod{q_i^{\alpha_i}}.$$

З умови задачі витікає:

$$a^{\frac{p-1}{q_i}} \equiv g^{\frac{x_0(p-1)}{q_i}} \pmod{p}.$$

За допомогою таблиці з першого кроку знаходимо  $x_0$ . Тоді має місце конгруенція:

$$(ag^{-x_0})^{\frac{p-1}{q_i^2}} \equiv g^{\frac{x_1(p-1)}{q_i}} \pmod{p}.$$

Аналогічно, знаходимо з таблиці значення  $x_1$ . У загальному випадку, значення  $x_j$  знаходиться з порівняння

$$(ag^{-x_0 - x_1 q_i - \dots - x_{j-1} q_i^{j-1}})^{\frac{p-1}{q_i^{j+1}}} \equiv g^{\frac{x_j(p-1)}{q_i}} \pmod{p}.$$

*Крок 3.* Знайшовши  $\log_g a \pmod{q_i^{\alpha_i}}$ ,  $i = 1, \dots, s$ , знаходимо  $\log_g a \pmod{p - 1}$ , що й буде нашою відповіддю, оскільки  $0 \leq \log_g a \pmod{p} < p - 1$ .

Обчислювальна складність алгоритму Поліга-Хеллмана є поліноміальною, а саме,  $O((\log p)^{c_1})$ , якщо всі прості дільники  $q_i$  числа  $p - 1$  не перевищують  $(\log p)^{c_2}$ , де  $c_1, c_2$  — додатні константи. Наприклад, це має місце для простих чисел вигляду  $p = 2^\alpha + 1$ . Якщо ж  $p - 1$  має простий дільник  $q$ ,  $q \geq p^c$ , де  $c > 0$ , то алгоритм Поліга-Хеллмана буде мати експоненційну складність. Іншими словами, всі дільники числа  $p - 1$  мають бути  $(\log p)^{c_2}$ -гладкими числами у сенсі наступного визначення: натуральне число  $n$  називається  $B$ -гладким, або гладким щодо межі  $B$ , якщо всі його прості дільники не перевищують  $B$ .

### 1.2.3. Алгоритм index-calculus

Ідею використання деякої факторної бази (тобто, набору невеликих, зазвичай простих чисел для яких порівняно швидко обчислюються значення їх логарифму) для знаходження дискретного логарифму було опрацьовано різними авторами. Однією з великих переваг використання факторної бази є те, що її логарифми достатньо обчислити один раз для заданої групи. Після такого попереднього обчислення значно пришвидшуються послідовні обчислення логарифмів від різних значень. Враховуючи те, що у багатьох криптосистемах група може приймати досить обмежену кількість значень, це відкриває можливості для пришвидшення атак на подібні системи.

Одним з найбільш ефективних методів, побудованих на таких ідеях, є алгоритми групи index-calculus, що був запропонований у 1986 році. Опишемо загальну його схему згідно з [5]:

*1 етап.* Вибір факторної бази, що складається з невеликого набору малих незалежних (часто простих) елементів групи

*2 етап.* Генерація ступенів генератора, що є гладкими відносно елементів цієї факторної бази

*3 етап.* Розв'язання відповідної системи лінійних відношень, отриманої на другому етапі, для обчислення логарифмів елементів факторної бази

*4 етап.* Використання отриманих значень для обчислення логарифму будь-якого елемента групи

Слід також відмітити, що всі етапи, крім останнього, ніяким чином не залежать від конкретного елемента, дискретний логарифм якого шукається. Таким чином, можливе попереднє їх обчислення для заданої групи і подальше використання для пришвидшення атак на відомі криптосистеми.

Алгоритм index-calculus є дуже гнучким, існує багато різних підходів до реалізації окремих його етапів. Найкращі реалізації алгоритму мають евристичну складність близько  $L_p[\frac{1}{3}, c]$ .

## 2. Огляд платформ хмарних обчислень

### 2.1. Методика

Однією з найбільш значущих переваг моделі хмарних обчислень є легкість надання та вивільнення обчислювальних ресурсів. Це дозволяє оперативно адаптувати систему до різних рівнів навантажень, економлячи кошти за низького та швидко адаптуючись до високого. Зважаючи на ці два крайні випадки, було проведено дослідження мінімальних та максимальних за обсягом обчислювальних ресурсів конфігурацій на трьох основних платформах: Google Cloud, Amazon AWS та Microsoft Azure (мотивацією вибору саме цих платформ є їх лідерські позиції, що підтверджуються, зокрема, експертами фірми Gartner [6]). Напрацюванням системою визначається та, що дозволяє точніший контроль за загальною обчислювальною потужністю системи вузлів, з меншою ціною за рівних інших показниках.

Табл. 1. Найслабші конфігурації

Платформа	Назва	Кількість ядер	Обсяг RAM, Гб	Вартість години, доларів США
Amazon	t2.nano	1	0,5	0,0065
Microsoft	A1	1	1,75	0,044
Google	n1-standard-1	1	3,75	0,015

Табл. 2. Найпотужніші конфігурації

Платформа	Назва	Кількість ядер	Обсяг RAM, Гб	Вартість години, доларів США
Amazon	c3.8xlarge	32	60	1,68
Microsoft	A11	16	56	1,56
Google	n1-highcpu-32	32	28,8	1,216

## 2.2. Результати

Для стандартизації підходу а також з огляду на поширеність, відкритість та ефективність можливих реалізацій алгоритмів, у якості операційної системи вузла для усіх платформ обрано Linux. У якості процесорів у переважній більшості локацій усі обрані платформи використовують процесори Intel Xeon сімейства E5. У табл. 1 та табл. 2 описані загальні характеристики (кількість ядер, кількість оперативної пам'яті, вартість за годину роботи), відповідно найменш та найбільш потужних конфігурацій на даних платформах з-поміж тих, у яких є хоча б один виділений потік виконання інструкцій процесора.

Як можна побачити, найбільш адаптованим до якнайточнішого виділення лише необхідних ресурсів є сервіс AWS EC2 компанії Amazon.

## Висновки

Розглянувши підходи до розв'язання проблеми дискретного логарифму з урахуванням можливості їх масової паралелізації, можна сказати, що одними з найшвидших сам по собі, а також зручними до розподілення на деякий масив обчислювальних вузлів задяки гнучкості окремих етапів, виявилися алгоритми групи index-calculus. Попередні результати та рекорди обчислення дискретного логарифму у своїй більшості використовують саме варіації цих методів.

З аналізу систем хмарних обчислень можна сказати, що платформа фірми Amazon має перевагу у гнучкості виділення необхідного обчислювального ресурсу. Загалом є доцільною побудова реалізації алгоритму index-calculus із задіянням великої кількості вузлів цієї платформи.

## Перелік використаних джерел

1. Mell Peter, Grance Tim. The NIST definition of cloud computing. — 2011.
2. McCurley Kevin S. The discrete logarithm problem // Proc. of Symp. in Applied Math. — Vol. 42. — 1990. — P. 49–74.
3. Shanks Daniel. The infrastructure of a real quadratic field and its applications // Proc. 1972 Number Theory Conf., Boulder, Colorado. — 1972. — P. 217–224.
4. Vasilenko Oleg Nikolaevich. Number-theoretic algorithms in cryptography. — American Mathematical Soc., 2007. — Vol. 232.
5. Schirokauer Oliver, Weber Damian, Denny Thomas. Discrete logarithms: the effectiveness of the index calculus method // Algorithmic number theory. — Springer, 1996. — P. 337–361.
6. Lydia Leong Douglas Toombs Bob Gill. Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. — 2015.