WIKIPEDIA

Discrete logarithm

In the mathematics of the real numbers, the logarithm $\log_b a$ is a number x such that $b^x = a$, for given numbers a and b. Analogously, in any group G, powers b^k can be defined for all integers k, and the **discrete logarithm** $\log_b a$ is an integer k such that $b^k = a$. In number theory, the more commonly used term is **index**: we can write $x = \mathrm{ind}_r\, a \pmod{m}$ (read the index of a to the base r modulo m) for $r^x \equiv a \pmod{m}$ if r is a primitive root of m and $\gcd(a,m) = 1$.

Discrete logarithms are quickly computable in a few special cases. However, no efficient method is known for computing them in general. Several important algorithms in public-key cryptography base their security on the assumption that the discrete logarithm problem over carefully chosen groups has no efficient solution.

Contents

Definition

Let G be any group. Denote its group operation by multiplication and its identity element by 1. Let b be any element of G. For any positive integer k, the expression b^k denotes the product of b with itself k times:

$$b^k = \underbrace{b \cdot b \cdots b}_{k \text{ factors}}.$$

Similarly, let b^{-k} denote the product of b^{-1} with itself k times. For $k = 0$, the kth power is the identity: $b^0 = 1$.

Let a also be an element of G. An integer k that solves the equation $b^k = a$ is termed a **discrete logarithm** (or simply **logarithm**, in this context) of a to the base b. One writes $k = \log_b a$.

Examples

Powers of 10

The powers of 10 form an infinite subset $G = \{\ldots, 0.001, 0.01, 0.1, 1, 10, 100, 1000, \ldots\}$ of the rational numbers. This set G is a cyclic group under multiplication, and 10 is a generator. For any element a of the group, one can compute $\log_{10} a$. For example, $\log_{10} 10000 = 4$, and $\log_{10} 0.001 = -3$. These are instances of the discrete logarithm problem.

Other base-10 logarithms in the real numbers are not instances of the discrete logarithm problem, because they involve non-integer exponents. For example, the equation $\log_{10} 53 = 1.724276\ldots$ means that $10^{1.724276\ldots} = 53$. While integer exponents can be defined in any group using products and inverses, arbitrary real exponents in the real numbers require other concepts such as the exponential function.

Powers of a fixed real number

A similar example holds for any non-zero real number b. The powers form a multiplicative subgroup $G = \{\ldots, b^{-3}, b^{-2}, b^{-1}, 1, b^1, b^2, b^3, \ldots\}$ of the non-zero real numbers. For any element a of G, one can compute $\log_b a$.

Modular arithmetic

One of the simplest settings for discrete logarithms is the group $(\mathbf{Z}_p)^\times$. This is the group of multiplication modulo the prime p. Its elements are congruence classes modulo p, and the group product of two elements may be obtained by ordinary integer multiplication of the elements followed by reduction modulo p.

The kth power of one of the numbers in this group may be computed by finding its kth power as an integer and then finding the remainder after division by p. When the numbers involved are large, it is more efficient to reduce modulo p multiple times during the computation. Regardless of the specific algorithm used, this operation is called modular exponentiation. For example, consider $(\mathbf{Z}_{17})^\times$. To compute 3^4 in this group, compute $3^4 = 81$, and then divide 81 by 17, obtaining a remainder of 13. Thus $3^4 = 13$ in the group $(\mathbf{Z}_{17})^\times$.

The discrete logarithm is just the inverse operation. For example, consider the equation $3^k \equiv 13 \pmod{17}$ for k. From the example above, one solution is $k = 4$, but it is not the only solution. Since $3^{16} \equiv 1 \pmod{17}$—as follows from Fermat's little theorem—it also follows that if n is an integer then $3^{4+16n} \equiv 3^4 \times (3^{16})^n \equiv 13 \times 1^n \equiv 13 \pmod{17}$. Hence the equation has infinitely many solutions of the form $4 + 16n$. Moreover, because 16 is the smallest positive integer m satisfying $3^m \equiv 1 \pmod{17}$, these are the only solutions. Equivalently, the set of all possible solutions can be expressed by the constraint that $k \equiv 4 \pmod{16}$.

Powers of the identity

In the special case where b is the identity element 1 of the group G, the discrete logarithm $\log_b a$ is undefined for a other than 1, and every integer k is a discrete logarithm for $a = 1$.

Properties

Powers obey the usual algebraic identity $b^{k+l} = b^k b^l$. In other words, the function

$$f : \mathbf{Z} \to G$$

defined by $f(k) = b^k$ is a group homomorphism from the integers $\mathbf{Z}$ under addition onto the subgroup H of G generated by b. For all a in H, $\log_b a$ exists. Conversely, $\log_b a$ does not exist for a that are not in H.

If H is infinite, then $\log_b a$ is also unique, and the discrete logarithm amounts to a group isomorphism

$$\log_b : H \to \mathbf{Z}.$$

On the other hand, if H is finite of size n, then $\log_b a$ is unique only up to congruence modulo n, and the discrete logarithm amounts to a group isomorphism

$$\log_b : H \to \mathbf{Z}_n,$$

where $\mathbf{Z}_n$ denotes the additive group of integers modulo n.

The familiar base change formula for ordinary logarithms remains valid: If c is another generator of H, then

$$\log_c a = \log_c b \cdot \log_b a.$$

Algorithms

The discrete logarithm problem is considered to be computationally intractable. That is, no efficient classical algorithm is known for computing discrete logarithms in general.

A general algorithm for computing $\log_b a$ in finite groups G is to raise b to larger and larger powers k until the desired a is found. This algorithm is sometimes called *trial multiplication*. It requires running time linear in the size of the group G and thus exponential in the number of digits in the size of the group. Therefore, it is an exponential-time algorithm, practical only for small groups G.

Unsolved problem in computer science:

? *Can the discrete logarithm be computed in polynomial time on a classical computer?*

(more unsolved problems in computer science)

More sophisticated algorithms exist, usually inspired by similar algorithms for integer factorization. These algorithms run faster than the naïve algorithm, some of them linear in the square root of the size of the group, and thus exponential in half the number of digits in the size of the group. However none of them run in polynomial time (in the number of digits in the size of the group).

- Baby-step giant-step
- Function field sieve
- Index calculus algorithm
- Number field sieve
- Pohlig–Hellman algorithm
- Pollard's rho algorithm for logarithms
- Pollard's kangaroo algorithm (aka Pollard's lambda algorithm)

There is an efficient quantum algorithm due to Peter Shor.[1]

Efficient classical algorithms also exist in certain special cases. For example, in the group of the integers modulo p under addition, the power b^k becomes a product bk, and equality means congruence modulo p in the integers. The extended Euclidean algorithm finds k quickly.

Comparison with integer factorization

While computing discrete logarithms and factoring integers are distinct problems, they share some properties:

- both are special cases of the hidden subgroup problem for finite Abelian groups,
- both problems seem to be difficult (no efficient algorithms are known for non-quantum computers),
- for both problems efficient algorithms on quantum computers are known,
- algorithms from one problem are often adapted to the other, and

- the difficulty of both problems has been used to construct various cryptographic systems.

Cryptography

There exist groups for which computing discrete logarithms is apparently difficult. In some cases (e.g. large prime order subgroups of groups $(\mathbf{Z}_p)^\times$) there is not only no efficient algorithm known for the worst case, but the average-case complexity can be shown to be about as hard as the worst case using random self-reducibility.

At the same time, the inverse problem of discrete exponentiation is not difficult (it can be computed efficiently using exponentiation by squaring, for example). This asymmetry is analogous to the one between integer factorization and integer multiplication. Both asymmetries (and other possibly one-way functions) have been exploited in the construction of cryptographic systems.

Popular choices for the group G in discrete logarithm cryptography are the cyclic groups $(\mathbf{Z}_p)^\times$ (e.g. ElGamal encryption, Diffie–Hellman key exchange, and the Digital Signature Algorithm) and cyclic subgroups of elliptic curves over finite fields (*see* elliptic curve cryptography).

While there is no publicly known algorithm for solving the discrete logarithm problem in general, the first three steps of the number field sieve algorithm only depend on the group G, not on the specific elements of G whose finite log is desired. By precomputing these three steps for a specific group, one need only carry out the last step, which is much less computationally expensive than the first three, to obtain a specific logarithm in that group.[2]

It turns out that much Internet traffic uses one of a handful of groups that are of order 1024 bits or less, e.g. cyclic groups with order of the Oakley primes specified in RFC 2409. The Logjam attack used this vulnerability to compromise a variety of Internet services that allowed the use of groups whose order was a 512-bit prime number, so called export grade.[2]

The authors of the Logjam attack estimate that the much more difficult precomputation needed to solve the discrete log problem for a 1024-bit prime would be within the budget of a large national intelligence agency such as the U.S. National Security Agency (NSA). The Logjam authors speculate that precomputation against widely reused 1024 DH primes is behind claims in leaked NSA documents that NSA is able to break much of current cryptography.[2]

References

1. Shor, Peter (1997). "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". *SIAM Journal on Computing*. **26** (5): 1484–1509. arXiv:quant-ph/9508027 (https://arxiv.org/abs/quant-ph/9508027). doi:10.1137/s0097539795293172 (https://doi.org/10.1137%2Fs0097539795293172). MR 1471990 (https://www.ams.org/mathscinet-getitem?mr=1471990).
2. Adrian, David; Bhargavan, Karthikeyan; Durumeric, Zakir; Gaudry, Pierrick; Green, Matthew; Halderman, J. Alex; Heninger, Nadia; Springall, Drew; Thomé, Emmanuel; Valenta, Luke; VanderSloot, Benjamin; Wustrow, Eric; Zanella-Béguelin, Santiago; Zimmermann, Paul (October 2015). "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice" (https://weakdh.org/imperfect-forward-secrecy.pdf) (PDF).

- Rosen, Kenneth H. *Elementary Number Theory and Its Application* (6th ed.). Pearson. p. 368. ISBN 978-0321500311.
- Weisstein, Eric W. "Discrete Logarithm" (http://mathworld.wolfram.com/DiscreteLogarithm.html). *MathWorld*. Wolfram Web. Retrieved 1 January 2019.

Further reading

- Richard Crandall; Carl Pomerance. Chapter 5, *Prime Numbers: A computational perspective*, 2nd ed., Springer.

- Stinson, Douglas Robert (2006), *Cryptography: Theory and Practice* (3rd ed.), London: CRC Press, ISBN 978-1-58488-508-5

Retrieved from "https://en.wikipedia.org/w/index.php?title=Discrete_logarithm&oldid=894167717"

This page was last edited on 2019-04-26, at 10:53:43.