Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
עברית
Nederlands
Edit links

Index calculus algorithm

From Wikipedia, the free encyclopedia

In computational number theory, the **index calculus algorithm** is a probabilistic algorithm for computing discrete logarithms. Dedicated to the discrete logarithm in $(\mathbb{Z}/q\mathbb{Z})^*$ where q is a prime, index calculus lead to a family of algorithms adapted to finite fields and to some families of elliptic curves. The algorithm collects relations among the discrete logarithms of small primes, computes them by a linear algebra procedure and finally expresses the desired discrete logarithm with respect to the discrete logarithms of small primes.

Contents [hide]

Description [edit]

Roughly speaking, the discrete log problem asks us to find an *x* such that $g^x \equiv h \pmod{n}$, where *g*, *h*, and the modulus *n* are given.

The algorithm (described in detail below) applies to the group $(\mathbb{Z}/q\mathbb{Z})^*$ where *q* is prime. It requires a *factor base* as input. This *factor base* is usually chosen to be the number −1 and the first *r* primes starting with 2. From the point of view of efficiency, we want this factor base to be small, but in order to solve the discrete log for a large group we require the *factor base* to be (relatively) large. In practical implementations of the algorithm, those conflicting objectives are compromised one way or another.

The algorithm is performed in three stages. The first two stages depend only on the generator *g* and prime modulus *q*, and find the discrete logarithms of a *factor base* of *r* small primes. The third stage finds the discrete log of the desired number *h* in terms of the discrete logs of the factor base.

The first stage consists of searching for a set of *r* linearly independent *relations* between the factor base and power of the generator *g*. Each relation contributes one equation to a system of linear equations in *r* unknowns, namely the discrete logarithms of the *r* primes in the factor base. This stage is embarrassingly parallel and easy to divide among many computers.

The second stage solves the system of linear equations to compute the discrete logs of the factor base. Although a minor computation compared to the other stages, a system of hundreds of thousands or millions of equations is a significant computation requiring large amounts of memory, and it is *not* embarrassingly parallel, so a supercomputer is typically used.

The third stage searches for a power *s* of the generator *g* which, when multiplied by the argument *h*, may be factored in terms of the factor base $g^s h = (-1)^{f_0} 2^{f_1} 3^{f_2} \cdots p_r^{f_r}$.

Finally, in an operation too simple to really be called a fourth stage, the results of the second and third stages can be rearranged by simple algebraic manipulation to work out the desired discrete logarithm $x = f_0 \log_g(-1) + f_1 \log_g 2 + f_2 \log_g 3 + \cdots + f_r \log_g p_r - s$.

The first and third stages are both embarrassingly parallel, and in fact the third stage does not depend on the results of the first two stages, so it may be done in parallel with them.

The choice of the factor base size *r* is critical, and the details are too intricate to explain here. The larger the factor base, the easier it is to find relations in stage 1, and the easier it is to complete stage 3, but the more relations you need before you can proceed to stage 2, and the more difficult stage 2 is. The relative availability of computers suitable for the different types of computation required for stages 1 and 2 is also important.

Applications in other groups [edit]

It is noteworthy that the lack of the notion of *prime elements* in the group of points on elliptic curves, makes it impossible to find an efficient *factor base* to run index calculus method as presented here in these groups. Therefore this algorithm is incapable of solving discrete logarithms efficiently in elliptic curve groups. However: For special kinds of curves (so called supersingular elliptic curves) there are specialized algorithms for solving the problem faster than with generic methods. While the use of these special curves can easily be avoided, in 2009 it has been proven that for certain fields the discrete logarithm problem in the group of points on *general* elliptic curves over these fields can be solved faster than with generic methods. The algorithms are indeed adaptations of the index calculus method.[1]

The algorithm [edit]

Input: Discrete logarithm generator *g*, modulus *q* and argument *h*. Factor base {−1,2,3,5,7,11,...,p_r}, of length *r*+1.
Output: *x* such that $g^x \equiv h \pmod q$.

```
• relations ← empty_list
• for k = 1, 2, ...
    • Using an integer factorization algorithm optimized for smooth numbers, try to factor $g^k \mod q$ (Euclidean residue) using the factor base, i.e. find $e_i$'s such that $g^k \mod q = (-1)^{e_0} 2^{e_1} 3^{e_2} \cdots p_r^{e_r}$
    • Each time a factorization is found:
        • Store k and the computed $e_i$'s as a vector $(e_0, e_1, e_2, \ldots, e_r, k)$ (this is a called a relation)
        • If this relation is linearly independent to the other relations:
            • Add it to the list of relations
            • If there are at least r+1 relations, exit loop
• Form a matrix whose rows are the relations
• Obtain the reduced echelon form of the matrix
    • The first element in the last column is the discrete log of −1 and the second element is the discrete log of 2 and so on
• for s = 0, 1, 2, ...
    • Try to factor $g^s h \mod q = (-1)^{f_0} 2^{f_1} 3^{f_2} \cdots p_r^{f_r}$ over the factor base
    • When a factorization is found:
        • Output $x = f_0 \log_g(-1) + f_1 \log_g 2 + \cdots + f_r \log_g p_r - s.$
```

Complexity [edit]

Assuming an optimal selection of the factor base, the expected running time (using L-notation) of the index-calculus algorithm can be stated as $L_n[1/2, \sqrt{2} + o(1)]$.

History [edit]

The first to discover the idea was Kraitchik in 1922.[2] After DLP became important in 1976 with the creation of the Diffie-Hellman cryptosystem, R. Merkle from Stanford University rediscovered the idea in 1977. The first publications came in the next two years from Merkle's colleagues.[3][4] Finally, Adleman optimized the algorithm and presented it in the form we know it today.[5]

The Index Calculus family [edit]

Index Calculus inspired a large family of algorithms. In finite fields $\mathbb{F}_q$ with $q = p^n$ for some prime p, the state-of-art algorithms are the Number Field Sieve for Discrete Logarithms, $L_q\left[1/3, \sqrt[3]{64/9}\right]$, when p is large compared to q, the function field sieve, $L_q\left[1/3, \sqrt[3]{32/9}\right]$, and Joux,[6] $L_q\left[1/4 + \epsilon, c\right]$ for $c > 0$, when p is small compared to q and the Number Field Sieve in High Degree, $L_q[1/3, c]$ for $c > 0$ when p is middle-sided. Discrete logarithm in some families of elliptic curves can be solved in time $L_q\left[1/3, c\right]$ for $c > 0$, but the general case remains exponential.

External links [edit]

- Discrete logarithms in finite fields and their cryptographic significance, by Andrew Odlyzko

- Discrete Logarithm Problem, by Chris Studholme, including the June 21, 2002 paper "The Discrete Log Problem".
- A. Menezes, P. van Oorschot, S. Vanstone (1997). *Handbook of Applied Cryptography*. CRC Press. pp. 107–109. ISBN 0-8493-8523-7.

Notes [edit]

1. ^ Diem, C (2010). "On the discrete logarithm problem in elliptic curves". *Compositio Mathematica*.
2. ^ M. Kraitchik, *Théorie des nombres*, Gauthier–Villards, 1922
3. ^ Pohlig, S. *Algebraic and combinatoric aspects of cryptography*. Tech. Rep. No. 6602-1, Stanford Electron. Labs., Stanford, Calif., Oct. 1977.
4. ^ M.E. Hellman and J.M. Reyneri, *Fast computation of discrete logarithms in GF* (q),Advances in Cryptology–Proceedings of Crypto, 1983
5. ^ L. Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, In 20th Annual Symposium on Foundations of Computer Science, 1979
6. ^ A. Joux, *A new index calculus algorithm with complexity* $L(1/4+o(1))$ *in very small characteristic* [1]

V · T · E	Number-theoretic algorithms [hide]
Primality tests	AKS TEST · APR TEST · Baillie–PSW · ECPP TEST · Elliptic curve · Pocklington · Fermat · Lucas · *LUCAS–LEHMER* · *LUCAS–LEHMER–RIESEL* · *PROTH'S THEOREM* · *PÉPIN'S* · Quadratic Frobenius test · Solovay–Strassen · Miller–Rabin
Prime-generating	Sieve of Atkin · Sieve of Eratosthenes · Sieve of Sundaram · Wheel factorization
Integer factorization	Continued fraction (CFRAC) · Dixon's · Lenstra elliptic curve (ECM) · Euler's · Pollard's rho · $p-1$ · $p+1$ · Quadratic sieve (QS) · General number field sieve (GNFS) · *Special number field sieve (SNFS)* · Rational sieve · Fermat's · Shanks' square forms · Trial division · Shor's
Multiplication	Ancient Egyptian · Long · Karatsuba · Toom–Cook · Schönhage–Strassen · Fürer's
Discrete logarithm	BABY-STEP GIANT-STEP · Pollard rho · Pollard kangaroo · POHLIG–HELLMAN · **Index calculus** · Function field sieve
Greatest common divisor	Binary · Euclidean · Extended Euclidean · Lehmer's
Modular square root	Cipolla · Pocklington's · Tonelli–Shanks
Other algorithms	Chakravala · Cornacchia · Integer relation · Integer square root · Modular exponentiation · Schoof's
Italics indicate that algorithm is for numbers of special forms · SMALLCAPS indicate a deterministic algorithm	

Categories: Group theory