Discrete Logarithm (1994; Shor)

Pranab Sen,
Tata Institute of Fundamental Research, Mumbai, India
www.tcs.tifr.res.in/~pgdsen

Index terms: Discrete logarithm problem, hidden subgroup problem, cryptography, quantum Fourier transform, quantum algorithms.

1 Synonyms
Logarithms in groups.

2 Problem definition
Given positive real numbers \( a \neq 1, b \), the logarithm of \( b \) to base \( a \) is the unique real number \( s \) such that \( b = a^s \). The notion of discrete logarithm is an extension of this concept to general groups.

Problem 1 (Discrete logarithm)

**INPUT:** Group \( G \), \( a, b \in G \) such that \( b = a^s \) for some positive integer \( s \).

**OUTPUT:** The smallest positive integer \( s \) satisfying \( b = a^s \), also known as the discrete logarithm of \( b \) to the base \( a \) in \( G \).

The usual logarithm corresponds to the discrete logarithm problem over the group of positive reals under multiplication. The most common case of discrete logarithm is when the group \( G = \mathbb{Z}_p^* \), the multiplicative group of integers between 1 and \( p - 1 \) modulo \( p \), \( p \) prime. Another important case is when the group \( G \) is the group of points of an elliptic curve over a finite field.

3 Key results

The discrete logarithm problem in \( \mathbb{Z}_p^* \), \( p \) prime as well as in the group of points of an elliptic curve over a finite field, is believed to be intractable for randomised classical computers. That is any, possibly randomised, algorithm for the problem running on a classical computer will take time that is super-polynomial in the number of bits required to describe an input to the problem. The best classical algorithm for finding discrete logarithms in \( \mathbb{Z}_p^* \), \( p \) prime is Gordon’s [4] adaptation of the number field sieve which runs in time \( \exp(O((1/3)\log p)^{1/3}(\log \log p)^{2/3})) \).
In a breakthrough result, Shor [9] gave an efficient quantum algorithm for discrete logarithm; his algorithm runs in time polynomial in the bit-size of the input.

**Result 1 ([9])** There is a quantum algorithm solving discrete logarithm on \( n \)-bit inputs in time \( O(n^3) \) with probability at least \( 3/4 \).

### 3.1 Description of the discrete logarithm algorithm

Shor’s algorithm for discrete logarithm [9] makes essential use of an efficient quantum procedure for implementing a unitary transformation known as the quantum Fourier transform. His original algorithm gave an efficient procedure for performing the quantum Fourier transform only over groups of the form \( \mathbb{Z}_r \), \( r \) a ‘smooth’ integer, but nevertheless, he showed that this itself sufficed to solve discrete logarithm in the general case. In this article however, a more modern description of Shor’s algorithm is given. In particular, a result by Hales and Hallgren [5] is used which shows that the quantum Fourier transform over any finite cyclic group \( \mathbb{Z}_r \) can be efficiently approximated to inverse exponential precision.

A description of the algorithm is given below. A general familiarity with quantum notation on the part of the reader is assumed. A good introduction to quantum computing can be found in the book by Nielsen and Chuang [8]. Let \((G, a, b, \bar{r})\) be an instance of the discrete logarithm problem, where \(\bar{r}\) is a supplied upper bound on the order of \(a\) in \(G\). That is, there exists a positive integer \(r \leq \bar{r}\) such that \(a^r = 1\). By using an efficient quantum algorithm for order finding also discovered by Shor [9], it can be assumed that the order of \(a\) in \(G\) is known, that is, the smallest positive integer \(r\) satisfying \(a^r = 1\). Shor’s order finding algorithm runs in time \(O((\log \bar{r})^3)\). Let \(\epsilon > 0\).

The discrete logarithm algorithm works on three registers, of which the first two are each \(t\)-qubits long where \(t := O(\log \bar{r} + \log(1/\epsilon))\), and the third register is big enough to store an element of \(G\). Let \(U\) denote the unitary transformation

\[
U : |x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus (b^x a^y)\rangle,
\]

where \(\oplus\) denotes bitwise XOR. Given access to a reversible oracle for group operations in \(G\), \(U\) can be implemented reversibly in time \(O(t^3)\) by repeated squaring.

Let \(\mathbb{C}[\mathbb{Z}_r]\) denote the Hilbert space of functions from \(\mathbb{Z}_r\) to complex numbers. The computational basis of \(\mathbb{C}[\mathbb{Z}_r]\) consists of the delta functions \(\{|l\rangle\}_{0 \leq l \leq r-1}\). Let \(\text{QFT}_{\mathbb{Z}_r}\) denote the quantum Fourier transform over the cyclic group \(\mathbb{Z}_r\) defined as the following unitary operator on \(\mathbb{C}[\mathbb{Z}_r]\):

\[
\text{QFT}_{\mathbb{Z}_r} : |x\rangle \mapsto r^{-1/2} \sum_{y \in \mathbb{Z}_r} e^{-2\pi i xy/r} |y\rangle.
\]

It can be implemented in quantum time \(O(t \log(t/\epsilon) + \log^2(1/\epsilon))\) up to an error of \(\epsilon\) using one \(t\)-qubit register [5]. Note that for any \(k \in \mathbb{Z}_r\), \(\text{QFT}_{\mathbb{Z}_r}\) transforms the state \(r^{-1/2} \sum_{x \in \mathbb{Z}_r} e^{2\pi i k x/r} |x\rangle\) to the state \(|k\rangle\). For any integer \(l\), \(0 \leq l \leq r - 1\), define

\[
|\hat{l}\rangle := r^{-1/2} \sum_{k=0}^{r-1} e^{-2\pi i k l/r} |a^k\rangle.
\]
Thus, the state in Step 1(c) of the above algorithm can be written as

\[ \sum_{x,y \in \mathbb{Z}_r} |x\rangle|y\rangle|b^x a^y\rangle. \]

The working of the algorithm is explained below. From equation 1, it is easy to see that

\[ |b^x a^y\rangle = r^{-1/2} \sum_{l=0}^{r-1} e^{2\pi i l (sx+y)/r} |\hat{l}\rangle. \]

Thus, the state in Step 1(c) of the above algorithm can be written as

\[ r^{-1} \sum_{x,y \in \mathbb{Z}_r} |x\rangle|y\rangle|b^x a^y\rangle = r^{-3/2} \sum_{l=0}^{r-1} \sum_{x,y \in \mathbb{Z}_r} e^{2\pi i l (sx+y)/r} |x\rangle|y\rangle|\hat{l}\rangle \]

\[ = r^{-3/2} \sum_{l=0}^{r-1} \left[ \sum_{x \in \mathbb{Z}_r} e^{2\pi i slx/r} |x\rangle \right] \left[ \sum_{y \in \mathbb{Z}_r} e^{2\pi ily/r} |y\rangle \right] |\hat{l}\rangle. \]

Now, applying QFT_{\mathbb{Z}_r} to the first two registers gives the state in Step 1(d) of the above algorithm. Measuring the first two registers gives \((sl \mod r, l)\) for a uniformly distributed \(l, 0 \leq l \leq r - 1\) in Step 1(e). By elementary number theory, it can be shown that if integers \(l_1, l_2\) are uniformly and independently chosen between 0 and \(l - 1\), they will be co-prime with constant probability. In that case, there will be integers \(k_1, k_2\) such that \(k_1 l_1 + k_2 l_2 = 1\), leading to the discovery of the discrete logarithm \(s\) in Step 3 of the algorithm with constant probability. Since actually speaking only an \(\epsilon\)-approximate version of QFT_{\mathbb{Z}_r} can be applied, \(\epsilon\) can be set to be a sufficiently small constant, and this will still give the correct discrete logarithm \(s\) in Step 3 of the algorithm with constant probability. The success probability of Shor’s algorithm for discrete logarithm can be boosted to at least \(3/4\) by repeating it a constant number of times.

### Algorithm 1 (Discrete logarithm)

**INPUT:** Elements \(a, b \in G\), a quantum circuit for \(U\), the order \(r\) of \(a\) in \(G\).

**OUTPUT:** The discrete logarithm \(s\) of \(b\) to the base \(a\) in \(G\).

**RUNTIME:** A total of \(O(t^3)\) operations including four invocations of QFT_{\mathbb{Z}_r} and one of \(U\).

**PROCEDURE:**

1. Repeat Steps (a)-(e) twice obtaining \((sl_1 \mod r, l_1)\) and \((sl_2 \mod r, l_2)\):

   - **Initialisation:**
     - (a) \(|0\rangle|0\rangle|0\rangle\)
     - (b) \(\mapsto r^{-1} \sum_{x,y \in \mathbb{Z}_r} |x\rangle|y\rangle|0\rangle\)
     - (c) \(\mapsto r^{-1} \sum_{x,y \in \mathbb{Z}_r} |x\rangle|y\rangle|b^x a^y\rangle\)
     - (d) \(\mapsto r^{-1/2} \sum_{l=0}^{r-1} |sl \mod r\rangle|l\rangle|\hat{l}\rangle\)
     - (e) \(\mapsto (sl \mod r, l)\)

2. If \(l_1\) is not co-prime to \(l_2\), abort.

3. Let \(k_1, k_2\) be integers such that \(k_1 l_1 + k_2 l_2 = 1\). Then, output \(s = k_1 (sl_1) + k_2 (sl_2) \mod r\).
3.2 Generalisations of the discrete logarithm algorithm

The discrete logarithm problem is a special case of a more general problem called the hidden subgroup problem [8]. The ideas behind Shor’s algorithm for discrete logarithm can be generalised in order to yield an efficient quantum algorithm for hidden subgroups in abelian groups (see e.g. [1] for a brief sketch). It turns out that finding the discrete logarithm of $b$ to the base $a$ in $G$ reduces to the hidden subgroup problem in the group $\mathbb{Z}_r \times \mathbb{Z}_r$ where $r$ is the order of $a$ in $G$. Besides discrete logarithm, other cryptographically important functions like integer factoring, finding the order of permutations as well as finding self-shift-equivalent polynomials over finite fields can be reduced to instances of hidden subgroup in abelian groups.

4 Applications

The assumed intractability of the discrete logarithm problem lies at the heart of several cryptographic algorithms and protocols. The first example of public-key cryptography, namely the Diffie-Hellman key exchange [2], uses discrete logarithms, usually in the group $\mathbb{Z}_p^*$ for a prime $p$. The security of the U. S. national standard Digital Signature Algorithm (see e.g. [7] for details and more references) depends on the assumed intractability of discrete logarithms in $\mathbb{Z}_p^*$, $p$ prime. The ElGamal public key cryptosystem [3] and its derivatives use discrete logarithms in appropriately chosen subgroups of $\mathbb{Z}_p^*$, $p$ prime. More recent applications include those in elliptic curve cryptography [6], where the group consists of the group of points of an elliptic curve over a finite field.

5 Cross references

Factoring (00002), Abelian Hidden Subgroup Problem (00004).

6 Recommended reading


