CS 812
Lecture 24
Monday 3/23/20

New Topic – Discrete Logarithms

Let G be a group, with $a, b \in G$. If there is an integer x such that $a^x = b$, we call x a *discrete logarithm* (base a) of b. The word *index* is also used.

Usually $|G| < \infty$, and we reduce x modulo the order m of a. The order is the least $m \geq 1$ such that $a^m = 1$, so that $0 \leq x < m$.

As the name suggests, the algebraic properties of discrete logs are similar to ordinary logarithms, so for example

$$\log_a(b_1 b_2) \equiv \log_a(b_1) + \log_a(b_2) \pmod{m}.$$

Exercise: Show that other familiar laws for logarithms hold for discrete logs.

Just as with ordinary logs, discrete logarithms were tabulated and used to simplify computations.

Jacobi's *Canon Arithmeticus* (1839) gave discrete logs for the multiplicative groups of prime powers up to 1000.

Western and Miller's *Tables of Indices and Primitive Roots* (1968) did the same for primes up to 50021.

How Hard is the Discrete Log?

The difficulty of solving $a^x = b$ depends on the group and how it is presented.

Example 1: $G = (\mathbf{Z}_n, +)$. Then the equation to solve is $ax \equiv b \pmod{n}$. When $a \in \mathbf{Z}_n^*$, we have $x \equiv a^{-1}b$, and we can use the extended Euclidean algorithm to find x in polynomial time.

Example 2: Let p be prime, and $G = (\mathbf{Z}_p^*, \cdot)$. As is known, G is cyclic of order $p - 1$.

The discrete log problem is easy when $p - 1$ is smooth.

When $p - 1$ has large prime factor, there is no obvious efficient algorithm for discrete logs. As we will see later, this problem has roughly the same complexity as factoring.

Example 3: The points on a smooth cubic defined over a finite field can be given a group structure. (In the trade, this is called an elliptic curve group.) When a is a group element of large prime order, the discrete log is thought to be harder than factoring integers of comparable size.

Brute Force Computation

We can always try to solve $a^x = b$ by brute force.

Try $a, a^2, a^3, \ldots$ until you hit b. In the worst case, this will use about $2m$ group operations (multiplication, comparison of elements).

Food for thought: Is this the best you can do, if the only operations allowed are multiplication and identity testing?

A Collision Algorithm

We'll now discuss a "collision" method that solves $a^x = b$ using about $\sqrt{m}$ operations.

First published by Daniel Shanks [Proc. AMS Symp. Pure Math. 20, 1971]. He called it the baby-step giant-step method, and the name stuck.

For simplicity assume that m, the order of a, is known. We want to solve $a^x = b$, with $0 \leq x < m$.

Method: Choose $r \geq \sqrt{m}$. If we knew x, we could split it in two by writing it in base r:

$$x = x_0 + x_1 r, \qquad 0 \leq x_i < r.$$

Then (using some algebra)

$$a^x = b \Longleftrightarrow a^{x_0} = b(a^{-r})^{x_1}.$$

Form two lists:

$$B = \{(a^{x_0}, x_0, 0) : 0 \leq x_0 < r\},$$
$$G = \{(b(a^{-r})^{x_1}, x_1, 1) : 0 \leq x_1 < r\}.$$

(The final bit, 0 or 1, just indicates which list a tuple came from.) Combine these lists and sort on the first entry, with the final bit as a secondary key. Two tuples with identical first entries and differing final entries give the collision

$$a^{x_0} = b(a^{-r})^{x_1},$$

so $x = x_0 + x_1 r$ is a discrete log of b as required.

Why must there be a collision? Imagine a circle with evenly spaced marks labeled $0, 1, \ldots, m-1$. The "baby steps" – from B – hit a block of r consecutive marks. The "giant steps" – from G – make skips of length r, so one of them must hit the block.

Run time analysis: It is natural to count group operations (multiplication, inverse, equality testing). To sort the tuples, however, we need to order the group elements. So we will assume that each group element is represented by a binary string, and to compare group elements, we just compare their strings. Such comparisons are also assigned unit cost.

We need the following:

a^{-1} – 1 operation.

a^{-r} – $O(\log r) = O(\log m)$ operations.

B – $O(r) = O(\sqrt{m})$ operations.

G – same.

sorted $B \cup G$ – $O(r \log r) = O(\sqrt{m} \log m)$ operations.

If the group is compactly represented, this algorithm uses $O(\sqrt{m} \log m)$ operations and $O(\sqrt{m} \log m)$ bits of space.

Factoring the Order

The following idea was published by Pohlig and Hellman [IEEE-IT, 1978].

Suppose we can factor m, the order of a:

$$m = q_1^{e_1} \cdots q_r^{e_r}, \qquad q_i \text{ distinct primes.}$$

Let us write C_n for a cyclic group of order n (that is, $\mathbf{Z}_n$ but written with multiplicative notation). Then $< a >$, the group generated by a, is isomorphic to C_m. By the Chinese remainder theorem, we have

$$C_m \cong C_{q_1^{e_1}} \times \cdots \times C_{q_r^{e_r}}.$$

The projection on the i-th factor is given by

$$c \mapsto c^{m/\prod_{j \neq i} q_j^{e_j}}.$$

Note: There is something subtle going on here. The usual way to do the projection would be to reduce the exponent x mod $q_i^{e_i}$. But we don't know x, so we have to be more devious. To fully understand this, you should consider a concrete example, such as $\mathbf{Z}_{15}$, with the map $x \mapsto (5x, 3x)$. Note that the image lives in the Cartesian product $\mathbf{Z}_{15} \times \mathbf{Z}_{15}$.

Applying this to both a and b, we get r separate discrete log problems, one for each factor. If these can be solved, we can use the Chinese remainder theorem to get a solution to $a^x = b$.

A further reduction is possible for factors of prime-power order. For C_{q^e}, we have the chain of subgroups

$$1 \subset C_q \subset C_{q^2} \subset \cdots \subset C_{q^e}.$$

Each factor group is cyclic of order q. To solve $a^x = b$ in C_{q^e}, we can write the unknown in base q, say

$$x = x_0 + x_1 q + \cdots x_{e-1} q^{e-1},$$

and then solve for $x_0, x_1, \ldots, x_{e-1}$ in turn.

In detail, suppose we want x_0. Take $a^x = b$ and raise both sides to the power q^{e-1}, to get

$$(a^{q^{e-1}})^x = (a^{q^{e-1}})^{x_0} = b^{q^{e-1}}.$$

This is a discrete log problem in a cyclic group of order q.

Having found x_0, we can now solve

$$(a^q)^{x_1+\cdots+x_{e-1}q^{e-1}} = ba^{-x_0},$$

which is the same kind of problem but with e reduced by 1.

Upshot: The time to solve $a^x = b$ is about $Q^{1/2}$, where Q is the largest prime divisor of the order of a.

We can use any multiple of the order in place of m. One traditional choice is the order of the group.

An Exercise for You

Choose a prime p for which $p-1$ is highly composite, such as 3823.

Find the least primitive root g for p.

Solve $g^x = g+1$ in $\mathbf{Z}_p^*$, using both the baby-step giant-step algorithm and the Pohlig-Hellman procedure. Make sure your answers agree.