Computing roots in finite fields

Arjen Stolk

Leiden, October 13, 2008

Problem 1.
Given an element $\alpha \in \mathbb{F}_q$ and a positive integer e, find $\beta \in \mathbb{F}_q$ such that $\beta^e = \alpha$.

Problem 2.
Given a finite cyclic group C of order n, an element $a \in C$ and a positive integer e, find $b \in C$ such that $b^e = a$.

Example.
Let n be a positive integer and consider $C = \mathbb{Z}/n\mathbb{Z}$. Let $a \in C$ and $e \in \mathbb{Z}_{>0}$. In this case our problem comes down to finding all b such that

$$be \equiv a \bmod n.$$

This can be done quickly using the Euclidean algorithm.

Let C be a finite cyclic group of order n. We assume that there is a total ordering on the elements of C. Furthermore, we assume that the following things can be done in polynomial time in $\log n$:

- given $a, b \in C$, computing ab;
- given $a \in C$, computing a^{-1};
- given $a, b \in C$, deciding if $a = b$;
- given $a, b \in C$, deciding if $a < b$;
- picking a uniform random element $a \in C$.

A useful building block for algorithms involving elements of C is the following.

Lemma.
Let $a, b \in C$ and $e \in \mathbb{Z}_{\geq 0}$ then the there is an algorithm that computes ab^e using $O(\log e)$ operations.

Algorithm 1.
Input: $a, b \in C$, $e \in \mathbb{Z}_{\geq 0}$
Output: ab^e

1. $x \leftarrow a$
2. while $e > 0$:
   - 2.1 if e is odd, then $x \leftarrow bx$
   - 2.2 $b \leftarrow b^2$
   - 2.3 $e \leftarrow \lfloor e/2 \rfloor$
3. output x

(skip to slide 9)

Suppose that the algorithm is called with inputs **a**, **b** and **e**. We claim that at the start of each iteration of 2, we have $xb^e = \mathbf{a}\mathbf{b}^{\mathbf{e}}$.

Suppose b is even. Let $x' = x$, $b' = b^2$ and $e' = e/2$.
Then we have

$$x'(b')^{e'} = x(b^2)^{e/2} = xb^e = \mathbf{a}\mathbf{b}^{\mathbf{e}}.$$

Suppose b is odd. Let $x' = bx$, $b' = b^2$ and $e' = (e-1)/2$.
Then we have

$$x'(b')^{e'} = xb(b^2)^{(b-1)/2} = xb^e = \mathbf{a}\mathbf{b}^{\mathbf{e}}.$$

After each iteration e has strictly decreased, so the algorithm stops with $e = 0$ and hence $x = xb^e = \mathbf{a}\mathbf{b}^{\mathbf{e}}$.

With every iteration, e is halved. Hence step 2 will be repeated at most $\lceil {}^2\log e \rceil$ times. Each iteration of step 2 takes at most 2 multiplications in C.

In total, this takes $O(\log e)$ operations.

Theorem.
Let n and e be coprime positive integers. Let C be a finite cyclic group of order n and let $a \in C$. Then there is an efficient algorithm to compute the unique $b \in C$ such that $b^e = a$.

Proof.
Suppose that z is a generator of C. Write $a = z^s$ and $b = z^t$. Then we are looking for t such that $z^{te} = z^s$. That is, we are looking for solutions of $te \equiv s \bmod n$. Using the Euclidean algorithm, compute p and q such that $pe + qn = 1$. Then $te \equiv s \bmod n$ if and only if $t \equiv ps \bmod n$. That is, $b = a^p$ is the unique solution.

Let m and n be coprime integers. Let C be a finite cyclic group of order mn. Write C_m and C_n for the unique subgroups of order m and n respectively.

Let p and q be integers such that $mp + nq = 1$. Then the inverse of the natural map $C_m \times C_n \to C$ is given by

$$\begin{array}{ccc} C & \longrightarrow & C_m \times C_n \\ x & \longmapsto & (a^{qn}, a^{pm}). \end{array}$$

Let e and k be positive integers. Let C be a finite cyclic group of order e^k.

Theorem.
Let z be a generator of C and $a \in C$, then there is an algorithm that computes $m \in \mathbb{Z}_{\geq 0}$ such that $a = z^m$ using $O(k^2 \sqrt{e} \log e)$ operations.

Let C be a finite group of order e and z a generator of C.

Lemma.
Let $a \in C$ then there is an algorithm that computes a positive integer m such that $a = z^m$ using $O(\sqrt{e} \log e)$ operations.

Algorithm 2.
Input: $a, z \in C$
Output: m such that $z^m = a$

1. $f \leftarrow \lceil \sqrt{e} \rceil$
2. for $i = 0, \ldots, f-1$:
   put $z_i \leftarrow z^{-i}$ and $Z_i = z^{fi}$
3. for $i = 0, \ldots, f-1$:
   check if there is a j such that $az_i = Z_j$
4. output $i + fj$

(skip to slide 15)

For the correctness, note that if $az_i = Z_j$, we have $az^{-i} = z^{fj}$, that is, $a = z^{i+fj}$.

To look for az_i inside $\{Z_0, \ldots Z_{f-1}\}$ using only $O(\log e)$ operations, we use the total ordering of elements of C, for example by storing the Z_j in a binary tree.

Algorithm 3.

Input: $a, z \in C$

Output: m such that $z^m = a$

1. $m \leftarrow 0$, $w \leftarrow z^{e^{k-1}}$, $r \leftarrow k-1$
2. while $a \neq 1$:
   - 2.1 $b \leftarrow a^{e^r}$
   - 2.2 using algorithm 2 on the subgroup generated by w, determine s such that $w^s = b$
   - 2.3 $a \leftarrow az^{-se^{k-r-1}}$, $m \leftarrow m + se^{k-r-1}$, $r \leftarrow r-1$
3. output m

(skip to slide 19)

Let $e = 5$, $k = 6$. Let C be a finite group of order e^k and z a generator of C. Let $a = z^{4321} = z^{114241_5}$.

i	r	m	a	b	s
1	5	0	z^{114241_5}	z^{100000_5}	1
2	4	1	z^{114240_5}	z^{400000_5}	4
3	3	41_5	z^{114200_5}	z^{200000_5}	2
4	2	241_5	z^{114000_5}	z^{400000_5}	4
5	1	4241_5	z^{110000_5}	z^{100000_5}	1
6	0	14241_5	z^{100000_5}	z^{100000_5}	1
7	-1	114241_5	z^0	-	-

Suppose that the algorithm is called with input $\mathbf{a}$. We claim that at the start of each iteration of 2, we have $a^{e^{r+1}} = 1$ and $az^m = \mathbf{a}$.

Before step 2.3, let $a' = az^{-se^{k-r-1}}$ and $m' = m + se^r$. As $b = w^s$ we have

$$(a')^{e^r} = (az^{-se^{k-r-1}})^{e^r} = bw^{-s} = 1$$

and

$$a'z^{m'} = (az^{-se^{k-r-1}})z^{m+se^{k-r-1}} = az^m = \mathbf{a},$$

so the same relations hold for the next iteration.

At the start of the $(k+1)$-th iteration, we have $r = -1$ and so $a = a^{e^{r+1}} = 1$ and the loop stops. At this point we have $z^m = az^m = \mathbf{a}$.

We analyse each step:

1 takes $O(k \log e)$ operations;

2 is iterated at most k times:

   2.1 takes $O(k \log e)$ operations;

   2.2 uses algorithm 2 applied to a group of e elements, this requires $O(\sqrt{e} \log e)$ operations;

   2.3 takes $O(k \log e)$ operations.

In total, the algorithm requires $O(k^2 \sqrt{e} \log e)$ operations.

Algorithms 2 and 3 require a generator of C to work. Finding these generators is the only non-deterministic part of the root finding algorithm.

Suppose that e is prime and $k \in \mathbb{Z}_{>0}$. Let C be a finite cyclic group of order e^k. Let $z \in C$. Then z is a generator of C if and only if $z^{e^{k-1}} \neq 1$.

A random element has a chance of $1/e$ of it *not* being a generator. Checking if an element is a generator takes $O(\log e)$ operations. So a generator can be found in expected $O(\log e)$ operations.

By combining the algorithms we have seen, we find a single algorithm to solve our original problem. This algorithm is due to Shanks.

Theorem.
Let C be a finite cyclic group of order n and e be a prime number. Then there is a probabilistic polynomial time algorithm that given $a \in C$ computes $b \in C$ such that $b^e = a$ or shows that no such b exists. The algorithm takes an expected $O((\log n)^2 \sqrt{e} \log e)$ group operations.

Algorithm 4.
Input: $a \in C$, e prime
Output: $b \in C$ such that $a = b^e$ or FAILURE if no such b exists

1. find k, m, d such that $n = e^k m$ and $md \equiv -1 \bmod e$
2. pick random elements x of C until $x^{n/e} \neq 1$
3. $z \leftarrow x^m$, $w \leftarrow x^{n/e}$, $f \leftarrow \lceil \sqrt{e} \rceil$
4. for $i = 0, \ldots, f-1$: put $w_i \leftarrow w^{-i}$, $W_i \leftarrow w^{fi}$
5. set $y \leftarrow z$, $r \leftarrow k$, $c \leftarrow a^{md}$, $b \leftarrow a^{(md+1)/e}$
6. while $c \neq 1$:
   - 6.1 find the smallest $s \geq 1$ such that $c^{e^s} = 1$
     if $s = r$ output FAILURE
   - 6.2 find i, j such that $c^{e^{s-1}} w_i = W_j$
   - 6.3 set $t \leftarrow y^{e^{r-s-1}}$, $y \leftarrow t^e$, $r \leftarrow s$, $c \leftarrow cy^{-i-fj}$, $b \leftarrow bt^{-i-fj}$
7. output b

All elements should be familiar from previous algorithms.

To convince you of the correctness of the algorithm, note that during every iteration of 6, we have the following:

- y is a generator of the subgroup of e^r elements;
- c is an element of that subgroup;
- $ac = b^e$.

The exponent r is decreasing, so eventually we will have $c = 1$ and then $a = b^e$.

As remarked before, the only non-deterministic step of algorithm 4 is finding the generator of the subgroup of order e^k (step 2).

In the case that $C = \mathbb{F}_p^\times$, we expect just trying 1, 2, 3, etc. as candidates should work fast enough. However, this has only been proved assuming the Riemann Hypothesis.