Lecture 6:

Lecture 6:
Discrete Logarithms

Objectives

- To examine algorithms for solving the discrete logarithm problem.
- To introduce the Pohlig-Hellman algorithm.
- To introduce the Baby-Step/Giant-Step algorithm.
- To explain the methods of Pollard.
- To show how discrete logarithms can be solved in finite fields using algorithms like those used for factoring.

Discrete Logarithms

- Solving the discrete logarithm problem,

$$h = g^x$$

  in various groups G.
- Two categories, either the algorithms are generic and apply to any finite abelian group or the algorithms are specific to the special group under consideration.

Pohlig-Hellman

- Pohlig-Hellman observation: the discrete logarithm problem in a group G is only as hard as the discrete logarithm problem in the largest subgroup of prime order in G.

Explaining Pohlig-Hellman

- Suppose we have a finite cyclic abelian group $G =< g >$, with order

$$N = \mathrm{ord}(g) = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}.$$

- $h \in G$ given such that $h = g^x$ for some unknown x.
- Idea: find x by first finding it modulo $p_i^{e_i}$ and then use CRT to recover it modulo N.

Explaining Pohlig-Hellman

- From basic group theory we know that there is a group isomorphism

$$\phi : G \to C_{p_1^{e_1}} \times \cdots \times C_{p_t^{e_t}},$$

  where $C_{p_i^{e_i}}$ are cyclic groups of order $p_i^{e_i}$.
- The projection of ϕ to the component $C_{p_i^{e_i}}$ is given by

$$\phi_{p_i} \begin{cases} G \to C_{p_i^{e_i}}, \\ f \mapsto f^{N/p_i^{e_i}}. \end{cases}$$

- The map ϕ_{p_i} is a group homomorphism so if we have $h = gx$ in G then we will have $\phi_{p_i}(h) = \phi_{p_i}(g)^x$ in $C_{p_i^{e_i}}$.
- But the discrete logarithm is only determined modulo $p_i^{e_i}$, so we get $x \bmod p_i^{e_i}$.
- Doing this for all primes would allow us to solve for x using CRT.

Explaining Pohlig-Hellman

Suppose we had some oracle $O(g, h, p, e)$ that solves discrete logarithm in $C_{p_i^{e_i}}$. We can then solve for x:

Algorithm 13.1: Algorithm to find DLP in group of order N, given an oracle for DLP for prime power divisors of N

$S = \{\}$
forall *primes* p *dividing* N **do**
    Compute the largest e such that $T = p^e$ divides N
    $g1 = g^{N/T}$
    $h1 = h^{N/T}$
    $z = O(g1, h1, p, e)$
    $S = S + \{(z, T)\}$
end
$x =$CRT(S)

Explaining Pohlig-Hellman

Solving the discrete logarithm problem in C_{p^e}: done by reducing to solving e discrete logarithm problems in the group C_p.

- Suppose $g, h \in C_{p^e}$ and that there is an x such that

$$h = g^x.$$

-

$$x = x_0 + x_1 p + x_2 p^2 + \cdots + x_{e-1} p^{e-1}.$$

- Use the following inductive procedure: Suppose we know x', the value of x modulo p^t, i.e.,

$$x' = x_0 + x_1 p + x_2 p^2 + \cdots + x_{t-1} p^{t-1}.$$

- We want to find x_t so we compute $x \bmod p^{t+1}$:

$$x = x' + p^t x''.$$

Explaining Pohlig-Hellman

- $$h = g^{x'}(g^{p^t})x''.$$
- Set $h' = hg^{-x'}$ and $g' = g^{p^t}$. Then we have
$$h' = g'^{x''}.$$
- Note that g' has order p^{e-t}. So the next step raises the above equation to the power $s = e - t - 1$. Set
$$h'' = h'^s, \quad g'' = g'^s.$$
- We finally reach a problem in C_p:
$$h'' = g''^{x_t}.$$

Example from book.

Baby-Step/Giant-Step Method

- a generic method which applies to any cyclic finite abelian group.
- Public cyclic group $G =< g >$, which we can now assume to have prime order p.
- fixed encoding of the elements of G, so in particular it is easy to store, sort and search a list of elements of G.

Baby-Step/Giant-Step Method

- We first write
$$x = x_0 + x_1\lceil\sqrt{p}\rceil.$$
Now, since $x \leq p$, we have that $0 \leq x0, x1 < \lceil\sqrt{p}\rceil$.
- Compute the Baby-Steps
$$g_i = g^i \text{ for } 0 \leq i < \lceil\sqrt{p}\rceil.$$
- Store
$$(g_i = g^i, i)$$
in a table sorted on the first entry.

Baby-Step/Giant-Step Method

- We now compute the Giant-Steps

$$h_j = hg^{-j\lceil\sqrt{p}\rceil} \text{ for } 0 \leq j < \lceil\sqrt{p}\rceil.$$

- We then try to find a match in the table of Baby-Steps, i.e. such that

$$g_i = h_j.$$

- If such a match occurs we have $x_0 = i$ and $x_1 = j$ since, if $g_i = h_j$,

$$g^i = hg^{-j\lceil\sqrt{p}\rceil}, \text{ and}$$

$$g^{i+j\lceil\sqrt{p}\rceil} = h.$$

PollardŠs Rho Algorithm.

- Baby-Step/Giant-Step method: $O(\sqrt{p})$ operations and memory.
- smaller space requirement can be achieved, if we accept that the running time is no longer deterministic.
- Suppose $f : S \rightarrow S$ is a random mapping.
- Pick a random value $x_0 \in S$ and compute

$$x_{i+1} = f(x_i) for i > 0.$$

The values $x_0, x_1, x_2,$ we consider as a deterministic random walk.

- Since S is finite we must eventually obtain

$$x_i = x_j$$

and so

$$x_{i+1} = f(x_i) = f(x_j) = x_{j+1}.$$

- The tail has expected length (i.e. the number of elements in the tail)

$$\sqrt{\pi n/8}$$

whilst the cycle has expected length (i.e. the number of elements in the cycle)

$$\sqrt{\pi n/8}$$

.

- The goal: find a collision in a random mapping, i.e. two values x_i and x_j with $i \neq j$ such that

$$x_i = x_j.$$

FloydŠs cycle finding algorithm:

- Removes the requirement of memory!
- we compute

$$(x_{i+1}, x_{2i+2}) = (f(x_i), f(f(x_{2i}))).$$

  We stop when we find

$$x_m = x_{2m}.$$

- Complexity estimate: $O(\sqrt{n})$.

Relating this to the discrete logarithm problem

- Partition the group into three sets S_1, S_2, S_3, and define

$$x_{i+1} = f(x_i) = \begin{cases} h \cdot x_i & x_i \in S_1 \\ x_i^2 & x_i \in S_2 \\ g \cdot x_i & x_i \in S_3 \end{cases}$$

- Keep track of three pieces of information

$$(x_i, a_i, b_i)$$

where

$$a_{i+1} = \begin{cases} a_i & x_i \in S_1 \\ 2a_i \bmod n & x_i \in S_2 \\ a_i + 1 \bmod n & x_i \in S_3 \end{cases}$$

$$b_{i+1} = \begin{cases} b_i + 1 \bmod n & x_i \in S_1 \\ 2b_i \bmod n & x_i \in S_2 \\ b_i & x_i \in S_3 \end{cases}$$

Relating this to the discrete logarithm problem

- Start with the triple

$$(x_0, a_0, b_0) = (1, 0, 0).$$

-

$$\log_g(x_i) = a_i + b_i \log_g(h) = a_i + b_i x.$$

A collision $x_m = x_{2m}$ gives

$$a_m + b_m x = a_{2m} + b_{2m} x,$$

and assuming $b_m \neq b_{2m}$, we obtain

$$x = \frac{a_{2m} - a_m}{b_m - b_{2m}} \bmod n.$$

Parallel Pollard's Rho

- In real life one uses a parallel version,
- Use of *distinguished points*.

Sub-exponential Methods for Finite Fields

Index-calculus algorithms