Lecture 8 Hardness of DLP §2.6 & 2.7

Recall that $DHP = EGP \leq DLP$.

For the rest will be discussing how hard DLP is.

Recall that for any group G, there is a discrete log problem:

DLP Let $g, h \in G$ where G is any group.

Let $N = ord(g)$. Find $x \in \mathbb{Z}/N\mathbb{Z}$ s.t. $g^x = h$.

The methods we give this week work for any group, so they also apply to Elliptic Curve DHP & EGP.

When we get to RSA we will discuss methods that don't apply to Elliptic Curve RSA.

There is a fairly strong correspondence between techniques for factoring & techniques for discrete log, so these general group discrete log techniques can be adopted to

General RSA (both conventional & elliptic curve RSA).
Similarly, the RSA $\mathbb{F}_p^\times$-specific algorithms can be adapted to discrete log.

Method 1 Brute force.

Compute $g^0, g^1, \dots, g^{N-1}$ until we find $g^x = h$

Worst case: Need $N-1$ multiplications.

If N has k bits, $k \sim \log_2(N)$ so we need roughly $N = O(2^k)$ operations.

Big O notation & Running time of Algorithms

$f = O(g)$ if $\frac{f(k)}{g(k)} \leq C$

For instance if $\lim_{k\to\infty} \frac{f(k)}{g(k)}$ exists & is $< \infty$,

$f = \Theta(g)$.

Ex $f(k) = k^n$, $g(k) = a^k$. $\lim_{k\to\infty} \frac{k^n}{a^k} = 0$

So $f(k) = O(g(k))$

Ex $\sin(k) = O(1)$ b/c $\frac{\sin(k)}{1} \leq 1$ is bounded.

Def $f(k) = o(g(k))$ if $g(k) = O(f(k))$.

(little o; Big O)

Def f has exponential growth if $\exists\ a, b > 1$ s.t. $f(k) = o(a^k)$ & $f(k) = O(b^k)$

(so f grows at least as fast as a^k & at most as fast as b^k)

Def f has subexponential growth if $f(k) = O(a^k)$ for all $a > 1$ & $f(k) = O(k^n)$ for all n.

(so f grows faster than any polynomial & slower than any exponential function)

Def f has polynomial growth if $f(k) = O(k^n)$ for some n.

We want to discuss the running time of algorithms.

Suppose we have an algorithm to compute something where if the input is size k the algorithm takes $f(k)$ steps.

- $f(k)$ exponential growth: called an "exponential time algorithm" Considered HARD.
- $f(k)$ subexponential growth: called a "subexponential time algorithm" Considered HARD
- $f(k)$ polynomial growth ~> "polynomial time algorithm" Considered EASY.

Rmk subexponential or exponential time only means asymptotically hard. Sometimes these work okay....

Rmk n^{1000} is asymptotically much smaller than 2^n but will likely still be impractically slow in practice... still considered "EASY"

Trivial DLP algorithm runs in $O(2^n)$ time. Slow! We will do a bit better.

§2.7 Shank's Babystep – Giantstep algorithm.

G a gp, $g \in G$ of order N, $h \in G$, want to solve $g^x = h$.

Algorithm

① Set $n = \lfloor \sqrt{N} \rfloor + 1$ e.g. $N=30$, $n = \lfloor \sqrt{30} \rfloor + 1 = 5 + 1 = 6$.

② Create two lists:

List 1: $g^0, g^1, \ldots, g^{n-1}$ (n numbers) babystep

List 2: $h, hg^{-n}, \ldots, hg^{-(n-1)n}$ (n numbers) giant step.

(3) Find the collision between the two lists (guaranteed). $g^i = h g^{-nj}$

(4) $h = g^{nj+i} \Rightarrow x = nj + i$.

Complexity

(2) $\sim 2n$ multiplications to make the two lists.

(3) To find collision, essentially have to sort lists $\rightarrow \Theta(n \log n) = \Theta(\sqrt{N} \log \sqrt{N})$

$= \Theta(2^{k/2} \cdot k/2)$

$= \Theta(k 2^{k/2})$.

Better than brute force, but still exponential time.

Ex In $\mathbb{F}_{31}^\times$, $g=3$, $\operatorname{ord}(g)=30$.

$h=10$.

Solve $3^x=10$ in $\mathbb{F}_{31}^\times$.

(1) $n=\lfloor\sqrt{30}\rfloor+1=6$

(2) List 1: g^0 g^1 g^2 g^3 g^4 g^5 g^6 g^{-6}

1 3 9 27 81 57 78 16^{-1}

19 26 16 2

List 2: h hg^{-6} hg^{-12} hg^{-18} hg^{-24} hg^{-30}

10 20 40 9

(3) $g^2=hg^{-12}$

(4) $h=g^{14}$ $\Rightarrow$ $x=14$.

Correctness of Algorithm

S'pose x satisfies $g^x = h$ & $0 \le x \le N-1$.

Write $x = nq + r$ $\quad 0 \le r < n$ bound on r.

Want bound on q, too.

$$q = \frac{x-r}{n} < \frac{N-r}{n} \le \frac{N}{n} < n$$

$n = \lfloor \sqrt{N} \rfloor + 1 > \sqrt{N} \Rightarrow N < n^2 \Rightarrow \frac{N}{n} < n.$

$$h = g^x = g^{nq+r} \iff h \cdot g^{-nq} = g^r$$

g^r is in list 1 b/c $0 \le r < n$.

hg^{-nq} is in list 2 b/c $0 \le q < n$.

$\Rightarrow$ must be a collision.