MPRI – Cours 2-12-2

F. Morain

Lecture II: discrete logarithm in generic groups

2011/10/11

Contents

I. Introduction

Def. (DLP) Given $G = \langle g \rangle$ of order n and $a \in G$, find $x \in [0..n[$ s.t. $a = g^x$.

Adaptive and non-adaptive: a is given beforehand, or only after some precomputation have been done (see Adleman's algorithm later).

Goal: find a resistant group.

Rem. DL is easy in $(\mathbb{Z}/N\mathbb{Z}, +)$, since $a = xg \bmod N$ is solvable in polynomial time (Euclid).

Relatively easy groups: (subexponential methods) finite fields, curves of very large genus, class groups of number fields.

Probably difficult groups: (exponential methods only?) elliptic curves.

Variants of the DL problem

Decisional DH problem: given (g, g^a, g^b, g^c), do we have $c = ab \bmod n$?

Computational DH problem: given (g, g^a, g^b), compute g^{ab}.

DL problem: given (g, g^a), find a.

Prop. DL $\Rightarrow$ CDH $\Rightarrow$ DCDH.

Thm. converse true for a large class of groups (Maurer & Wolf – see Smith's part).

More problems: ℓ-SDH (given g, g^{α}, ..., $g^{\alpha^{\ell}}$, compute $g^{\alpha^{\ell+1}}$).

Rem. Generalized problems on pairings.

Generic groups

Rem. generic means we cannot use specific properties of G, just group operations.

Known *generic* solutions:

- enumeration: $O(n)$;
- Shanks: deterministic time and space $O(\sqrt{n})$;
- Pollard: probabilistic time $O(\sqrt{n})$, space $O(1)$ elements of G.

Rem. All these algorithms can be more or less distributed.

II. The Pohlig-Hellman reduction

Idea: reduce the problem to the case n prime.

$$n = \prod_i p_i^{\alpha_i}$$

Solving $g^x = a$ is equivalent to knowing $x \bmod n$, i.e. $x \bmod p_i^{\alpha_i}$ for all i (chinese remainder theorem).

Idea: let $p^\alpha \mid\mid n$ and $m = n/p^\alpha$. Then $b = a^m$ is in the cyclic group of ordre p^α generated by g^m. We can find the log of b in this group, which yields $x \bmod p^\alpha$.

Cost: $O(\max(DL(p^\alpha))) = O(\max(DL(p)))$.

Consequence: in DH, n must have at least one large prime factor.

III. Baby steps giant steps (1/2)

Shanks:

$$x = cu + d, 0 \le d < u, \quad 0 \le c < n/u$$

$$g^x = a \Leftrightarrow a(g^{-u})^c = g^d.$$

Step 1 (baby steps): compute $\mathcal{B} = \{g^d, 0 \le d < u\}$;

Step 2 (giant steps):

- compute $f = g^{-u} = 1/g^u$;
- $h = a$;
- for $c = 0..n/u$
  {will contain af^c}
  if $h \in \mathcal{B}$ then stop; else $h = h \cdot f$.

End: $h = af^c = g^d$ hence $x = cu + d$.

Number of group operations: $C_o = u + n/u$, minimized for $u = \sqrt{n}$.

Shanks (2/2)

In the worst case, Step 2 requires n/u membership tests.

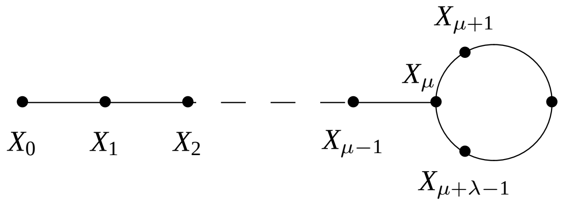
$\mathcal{B}$	insertions	membership tests
list	$u \times O(1)$	$\frac{n}{u}\ O(u)$
sorted	$O(u \log u)$	$\frac{n}{u}\ O(\log u)$
hash table	$u \times O(1)$	$\frac{n}{u}\ O(1)$

Prop. If membership test $= O(1)$, then dominant term is C_o, minimal for $u = \sqrt{n} \Rightarrow$ (deterministic) time and space $O(\sqrt{n})$.

Rem. all kinds of trade-offs possible if low memory available.

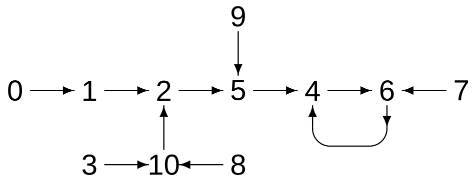
IV. Pollard's ρ

Prop. Let $f : E \rightarrow E$, $\#E = m$; $X_{n+1} = f(X_n)$ with $X_0 \in E$. The functional digraph of X is:



Ex1. If $E_m = G$ finite group with m elements, and $a \in G$ of ordre N, $f(x) = ax$ and $x_0 = a$, (x_n) purely periodic, i.e., $\mu = 0$, and $\lambda = N$.

Ex2. $E_m = \mathbb{Z}/11\mathbb{Z}$, $f : x \mapsto x^2 + 1 \bmod 11$:



Epact

Thm. (Flajolet, Odlyzko, 1990) When $m \rightarrow \infty$

$$\overline{\lambda} \sim \overline{\mu} \sim \sqrt{\frac{\pi m}{8}} \approx 0.627\sqrt{m}.$$

Prop. There exists a unique $e > 0$ (epact) s.t. $\mu \le e < \lambda + \mu$ and $X_{2e} = X_e$. It is the smallest non-zero multiple of λ that is $\ge \mu$: if $\mu = 0$, $e = \lambda$ and if $\mu > 0$, $e = \lceil \frac{\mu}{\lambda} \rceil \lambda$.

Thm. $\overline{e} \sim \sqrt{\frac{\pi^5 m}{288}} \approx 1.03\sqrt{m}$.

Floyd's algorithm:

```
X <- X0; Y <- X0; e <- 0;
repeat
    X <- f(X); Y <- f(f(Y)); e <- e+1;
until X = Y;
```


Application to the discrete log (à la Teske)

Compute the DL of $h = g^x$:

- Choose $y_0 = g^{\alpha_0} h^{\beta_0}$ for $\alpha_0, \beta_0 \in_R [0..n[$;
- Use a function F s.t. given $y = g^\alpha h^\beta$, one can compute efficiently $F(y) = g^{\alpha'} h^{\beta'}$;
- Compute the sequence $y_{k+1} = F(y_k)$ and the exponents $y_k = g^{\alpha_k} h^{\beta_k}$ until $y_i = y_j$.

When $y_i = y_j$, one gets

$$\alpha_i + \beta_i x \equiv \alpha_j + \beta_j x \bmod n$$

or

$$x \equiv (\alpha_j - \alpha_i)(\beta_i - \beta_j)^{-1} \bmod n$$

(with very high probability $\gcd(\beta_i - \beta_j, n) = 1$).

Two versions

Storing a few points:

- Compute r random points $M_k = g^{\gamma_k} h^{\delta_k}$ for $1 \le k \le r$;
- use $\mathcal{H} : G \rightarrow \{1, \ldots, r\}$;
- define $F(Y) = Y \cdot M_{\mathcal{H}(Y)}$.

Experimentally, $r = 20$ is enough to have a large mixing of points. Under a plausible model, this leads to a $O(\sqrt{n})$ method (see Teske).

Storing a lot of points:
(van Oorschot and Wiener)
Say a distinguished has some special form; we can store all of them to speed up the process.

V. Nechaev/Shoup theorem (à la Stinson)

Encoding function: injective map $\sigma : \mathbb{Z}/n\mathbb{Z} \rightarrow S$ where S is a set of binary strings s.t. $\#S \geq n$.

Ex. $G = (\mathbb{Z}/q\mathbb{Z})^* = \langle g \rangle$, $n = q - 1$, $\sigma : x \mapsto g^x \bmod q$, S can be $\{0,1\}^\ell$ where $q < 2^\ell$.

Wanted: a generic algorithm should work for any σ, in other words it receives σ as an input.

Oracle $\mathcal{O}$: given $\sigma(i)$ and $\sigma(j)$, computes $\sigma(ci \pm dj \bmod n)$ for any given known integers c and d. This is the only operation permitted.

Game: given $\sigma_1 = \sigma(1)$ and $\sigma_2 = \sigma(x)$ for random x, GENLOG succeeds if it outputs x.

Ex. Pollard's algorithm belongs to this class.

Reference: *Cryptography, Theory and Practice*, 2nd edition.

Stinson (2/5)

GENLOG produces $(\sigma_1, \sigma_2, \ldots, \sigma_m)$ using $\mathcal{O}$ where

$$\sigma_i = \sigma(c_i + x d_i \bmod n),$$

with $(c_1, d_1) = (1, 0)$ and $(c_2, d_2) = (0, 1)$, $(c_i, d_i) \in \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$.

Two cases: non-adaptive (choose c_i, d_i before receiving $\sigma(x)$) or adaptive.

Thm. Let $\beta = \mathrm{Proba}(\textit{GenLog}$ succeeds). For $\beta > \delta > 0$, one must have $m = \Omega(n^{1/2})$.

Stinson (3/5)

The non-adaptive case:

Step 1: (precomputations) GenLog chooses

$$\mathcal{C} = \{(c_i, d_i), 1 \leq i \leq m\} \subset \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$$

Step 2: upon receiving $\sigma(x)$, computes all $\sigma_i = \sigma(c_i + x d_i)$.

Step 3: check whether $\sigma_i = \sigma_j$ for some (i, j); since σ is injective, $\sigma_i = \sigma_j$ iff $c_i + x d_i \equiv c_j + x d_j$, return x.

Step 4: return a random value y.

Stinson (4/5)

Analysis:

$$\mathrm{Good}(\mathcal{C}) = \{(c_i - c_j)/(d_i - d_j)\}, \#\mathrm{Good}(\mathcal{C}) = \mathcal{G} \leq m(m-1)/2.$$

If $x \in \mathrm{Good}(\mathcal{C})$, GenLog returns x, otherwise some y.

α is the event "$x \in \mathrm{Good}(\mathcal{C})$":

$$\begin{aligned}
\mathrm{Proba}(\beta) &= \mathrm{Proba}(\beta \| \alpha)\mathrm{Proba}(\alpha) + \mathrm{Proba}(\beta \| \overline{\alpha})\mathrm{Proba}(\overline{\alpha}) \\
&= 1 \times \frac{\mathcal{G}}{n} + \frac{1}{n - \mathcal{G}} \times \frac{n - \mathcal{G}}{n} \\
&= \frac{\mathcal{G} + 1}{n} \leq \frac{m(m-1)/2 + 1}{n}.
\end{aligned}$$

$\Rightarrow$ if proba $> \delta > 0$, then m must be $\Omega(n^{1/2})$. $\square$

Stinson (5/5)

The adaptive case: For $1 \leq i \leq m$, $\mathcal{C}_i = \{\sigma_j, 1 \leq j \leq i\}$. Then a can be computed at time i if $a \in \text{Good}(\mathcal{C}_i)$. If $a \notin \text{Good}(\mathcal{C}_i)$, then $a \in \mathbb{Z}/n\mathbb{Z} - \text{Good}(\mathcal{C}_i)$ with proba $1/(n - \#\text{Good}(\mathcal{C}_i))$.

And now, what? this result tells you (only) that if you want an algorithm that is faster than Pollard's ρ or Shanks, then you have to work harder…