Discrete Logs for Hyperelliptic Curves

Summer School on Elliptic and Hyperelliptic Curve Cryptography

Nicolas Thériault
ntheriau@fields.utoronto.ca

Fields Institute

Discrete Logarithms

Suppose that $G = \langle a \rangle$, an additive group of order N, and $b \in G$.

The discrete logarithm of b in base a, $DL_a(b)$ is the smallest integer $\lambda \geq 0$ such that

$$b = [\lambda]a \quad .$$

The discrete log satisfies (for $a, b, c \in G$ and $k \in \mathbb{Z}$):

$$\begin{aligned} DL_a(b+c) &\equiv DL_a(b) + DL_a(c) \bmod N \\ DL_a([k]b) &\equiv kDL_a(b) \bmod N \\ DL_a(b) &\equiv DL_c(b)/DL_c(a) \bmod N \end{aligned}$$

Note: for the last relation, we assume that $a \in \langle c \rangle$.

The Discrete Log Problem

In generic groups, we have three square-root methods to compute $DL_a(b)$, which take $O\left(\sqrt{\text{group order}}\right)$ group operations:

- Baby Step - Giant Step (Shanks)
- Pollard ρ
- Pollard kangaroo

and one more method to take advantage of the prime decomposition of the group order:

- Pohlig-Hellman

Hyperelliptic Curves

For hyperelliptic curves (HEC) of genus g over the field $\mathbb{F}_q$, the order of the divisor class group is

$$q^g + O\left(gq^{g-1/2}\right) \quad .$$

To have a group of size N, we need $\log q \approx \frac{1}{g} \log N$.

For HECC, the cost of field arithmetic is $O((\log q)^2)$.

The group operation is done using Cantor's algorithm, which takes $O(g^2)$ field operations.

Looking quickly, the cost of a group operation seems to be stable if we fix a group order and vary the genus...

Hyperelliptic curves

If groups obtained from HEC are generic groups, then to have the same security as an EC over a field of 160 bits, a genus 5 curve needs a field of 32 bits...

At the 32 bit size we get a big boost in performance (on 32-bit processors), so genus 5 could be much faster!

But...

- We are applying asymptotic results to (small) fixed values (the conclusions could be wrong).
- We are assuming that divisor class groups are generic groups (hum... not really)

Index Calculus

Suppose that we have $p_1, p_2, \ldots, p_k \in G$ (a factor base).

Suppose that we know $DL_a(p_1), DL_a(p_1), \ldots, DL_a(p_k)$.

Suppose that we are able to write smooth relations

$$[\gamma]b = [\alpha_1]p_1 + [\alpha_2]p_2 + \ldots + [\alpha_k]p_k \quad .$$

Then

$$\gamma DL_a(b) \equiv \alpha_1 DL_a(p_1) + \alpha_2 DL_a(p_2) + \ldots + \alpha_k DL_a(p_k) \bmod N \quad ,$$

and if $\gcd(\gamma, N) = 1$, we get

$$DL_a(b) \equiv \frac{\alpha_1 DL_a(p_1) + \alpha_2 DL_a(p_2) + \ldots + \alpha_k DL_a(p_k)}{\gamma} \bmod N \quad .$$

How to find $DL_a(p_j)$

Look for random multiples of a that can be "factored" in terms of the p_j's, i.e.

$$[\beta_i]a = [\delta_{i,1}]p_1 + [\delta_{i,2}]p_2 + \ldots + [\delta_{i,k}]p_k \quad .$$

Each "factorization" gives a linear equation of the form

$$\beta_i = \delta_{i,1}DL_a(p_1) + \delta_{i,2}DL_a(p_2) + \ldots + \delta_{i,k}DL_a(p_k) \quad ,$$

where the $DL_a(p_j)$ are "variables".

Once we have a system of rank k, try to solve it. There is a solution since $p_j \in \langle a \rangle$ (for every j), and it must be unique since we have a system of rank k in k variables.

Index Calculus

We now have three problems to work out:

- How to choose the factor base
  - Prime divisors
- How to find smooth relations
  - Factorization
- How to solve a system of linear equations
  - Gaussian elimination, $O(k^3)$ operations mod N
  - Sparse linear algebra solvers, $O(\omega k^2)$
    - ω is the average number of non-zero coefficients per equation (small)
    - Lanczos' Algorithm
    - Wiedemann's algorithm

Variations

1. Find a smooth relation from $[\alpha]a$, one from $[\beta]b$ and "enough" relations of the form

$$[\gamma_{i,1}]p_1 + [\gamma_{i,2}]p_2 + \ldots + [\gamma_{i,k}]p_k = \mathbf{0} \quad .$$

   The smooth relations for $\mathbf{0}$ link the p_j's together (in a lattice). They can be used to write $[\beta]b$ in terms of $[\alpha]a$.

2. Find relations of the form

$$[\alpha_i]a + [\beta_i]b = [\delta_{i,1}]p_1 + [\delta_{i,2}]p_2 + \ldots + [\delta_{i,k}]p_k$$

   and find a linear combination for which the $\delta_{i,j}$'s are congruent to 0 mod N.
   This is the kernel approach.

The Kernel Approach

We have t "random" linear combinations

$$[\alpha_i]a + [\beta_i]b = \sum_{j=1}^{k} [\delta_{i,j}]p_j.$$

We can write the $\delta_{i,j}$'s in a matrix $M = (\delta_{i,j})$ over $\mathbb{Z}/N\mathbb{Z}$.

If $t \geq k+1$, the rank of the matrix must be smaller than the number of equations, so there exists a non-zero vector $\gamma = (\gamma_i)$ in the kernel of M, i.e. such that for every j

$$\sum_{i=1}^{t} \gamma_i \delta_{i,j} \equiv 0 \bmod N.$$

This gives us

$$
\begin{aligned}
0 &= \sum_{j=1}^{k} \left[\sum_{i=1}^{t} \gamma_i \delta_{i,j} \right] p_j \\
&= \sum_{i=1}^{t} \gamma_i \left(\sum_{j=1}^{k} [\delta_{i,j}] p_j \right) \\
&= \sum_{i=1}^{t} \gamma_i \left([\alpha_i] a + [\beta_i] b \right) \\
&= \left[\sum_{i=1}^{t} \gamma_i \alpha_i \right] a + \left[\sum_{i=1}^{t} \gamma_i \beta_i \right] b \\
&= [\alpha] a + [\beta] b
\end{aligned}
$$

The Kernel Approach

Advantages:

- Requires exactly $k+1$ relations (the other methods require more on average)
- The linear algebra is slightly faster.
- p_j does not have to be in $\langle a \rangle$ (we never compute $DL_a(p_j)$).

Inconvenient:

- The linear algebra must be restarted for every new discrete log in the group (if the $DL_a(p_j)$'s are known we only need to find one smooth relation with the new b).

Choosing the p_j's

A prime divisor is a semi-reduced divisor that cannot be written as the sum of two (or more) semi-reduced divisors except **0** and itself.

A prime divisor D can be written as

$$D = \sum_{j=0}^{i-1} \sigma^j(P) - iP_\infty$$

where P is a point in $C(\mathbb{F}_{q^i})$ (but not over any subfield) and σ is the Frobenius map over $\mathbb{F}_q$.

Every semi-reduced divisor "factors" uniquely as a sum of prime divisors

Remark: That's not true for divisor classes!

Choosing the p_j's

This is easier in the ideal class group...

A prime ideal is an ideal that cannot be written as a product of two ideals other than (1) and itself.

Prime ideals can be written in the form $(u(x), y - v(x))$ with $u(x)$ irreducible over $\mathbb{F}_q[x]$ and $\deg(v) < \deg(u)$.

The factorization of the ideal $(u(x), y - v(x))$ can be found by factoring $u(x)$. We get

$$(u(x), y - v(x)) = \prod_i (u_i(x), y - v_i(x))$$

with $u(x) = \prod_i u_i(x)$ and $v_i(x) \equiv v(x) \bmod u_i(x)$.

Factor Base

The size of a prime ideal $(u(x), y - v(x))$ is the degree of $u(x)$.

We let the factor base $\mathcal{B}$ be the set of all prime ideals of size at most B.

An ideal is B-smooth if it factors into prime ideals of size at most B, i.e. if all the irreducible factors of $u(x)$ are of degree at most B.

To choose the value of B we need to know how it affects finding B-smooth relations.

Note: $k_B = |\mathcal{B}| = |\{\text{prime divisors of size} \leq B\}|$

Probabilities

If smooth divisors (ideals) appear with probability p_B, how many divisors should we look at to be almost certain to find $k_B + 1$ smooth divisors?

Let $X_i = 1$ if the i^{th} divisor is smooth, 0 otherwise. X_i follows a Bernoulli distribution with probability p.

Let $Y_j = \sum_{i=1}^{j} X_i$. Since the X_i's are (assumed to be) independent, this is a Binomial distribution $B(j, p)$.

$$\begin{aligned} E[Y_j] &= jp_B \\ Var(Y_j) &= jp_B(1-p_B) \\ \sigma(Y_j) &< \sqrt{jp_B} \end{aligned}$$

We will need $k_B + 1$ smooth relations for some large k_B.
To have $E[Y_j] \approx k_B$, we need $j \approx k_B/p_B$.
But that's an expected value, we could end up short, or with too many... Can we be more precise?

Chebyshev's inequality:

$$Pr\left(\left|Y_j - E[Y_j]\right| \geq c\sigma(Y_j)\right) \leq 1/c^2$$

Example: 99.99% of the time we will get $k_B + 1$ smooth relations in less than $1.02k_B/p_B$ divisors if $k > 10^5$.
(This is much better than what we could say for Pollard Rho).

Index Calculus

If our factor base is bounded at size B, then we need to look at $O(k_B/p_B)$ divisors to have enough smooth relations.

Each divisor takes a group operation and a B-factorization ($O(g^2(\log q)^2)$ and $O(B^2g^2(\log q)^3)$ bit operations).

Solving the linear algebra problem takes $O(g{k_B}^2)$ operations mod N, each taking $O(g^2(\log q)^2)$ bit operations (since $N = O(q^g)$).

If we forget the "log terms", we get $O(k_B/p_B) + O({k_B}^2)$.

To minimize, we try to get the two terms to the same size.

Index Calculus

Using smooth relations in the class of the divisor **0**, Adleman, DeMarrais and Huang showed how to get

$$L_{q^g}(1/2, 4.36\ldots + o(1))$$

when $\log q \leq (2g)^{1-\varepsilon}$ (note: no sparse linear algebra).

Using the kernel approach, and tighter heuristics on p_B and k_B (by Enge and Stein), Enge and Gaudry found

$$L_{q^g}\left(1/2, \sqrt{2}\left(\sqrt{1+\frac{1}{2\nu}}+\sqrt{\frac{1}{2\nu}}\right)+o(1)\right)$$

when $\frac{g}{\log q} \geq \nu \geq 1$.

Finding Smooth Relations

We want to look at "randomly" chosen divisors.

If we look at divisors in the class zero, we can pick random principal divisors of the form $(A(x)y - B(x))$.
But how do we factor this, we are missing $u(x)$?

We are looking at ideals of the ring $\mathbb{F}_q[x,y]/(y^2 + h(x)y - f(x))$, so $(A(x)y - B(X))$ "contains" $R(x,y) = A(x)^2(y^2 + h(x)y - f(x))$ and we find

$$\begin{aligned} R(x,y) &= (A(x)y)^2 + h(x)A(x)(A(x)y) - f(x)A(x)^2 \\ &\equiv B(x)^2 + h(x)A(x)B(x) - f(x)A(x)^2 \bmod A(x)y - B(x) \\ &= u(x) \in (A(x)y - B(x)) \end{aligned}$$

so

$$(A(x)y - B(x)) = (u(x), y - (B(x)/A(x) \bmod u(x)))$$

Finding Smooth Relations

For the kernel method, we look at smooth $[\alpha]a + [\beta]b$.
We can find those using a random (or pseudo-random walk), just as we did with Pollard ρ. Instead of looking for distinguished points, we are looking for B-smooth divisors.

But we want to go much faster than Pollard ρ, so we don't really care about going back to the same smooth divisor.

This means we can remove the "function" part of the random map, i.e. we get

$$F(x) = x + ([\alpha_i]a + [\beta_i]b)$$

where $([\alpha_i]a + [\beta_i]b)$ is chosen at random (without any link to x) from a set of precomputed values.