Joux’s Recent Index Calculus Results
Part II: The Function Field Sieve and Joux’s Improvements

Joshua E. Hill
Department of Mathematics, University of California, Irvine

Cryptography Seminar
June 4, 2013
http://bit.ly/129p1If
v1.0
Talk Outline

1. Introduction

2. Modern Approaches to the Discrete Logarithm Problem

3. Conclusion, Mk. II
Introduction Outline

1. Introduction
   - From Part I
   - The Current State of Affairs

2. Modern Approaches to the Discrete Logarithm Problem

3. Conclusion, Mk. II
Subsection 1
Discrete and Discreet

Definition

Given a finite group $G$ (written multiplicatively), and a generator $g \in G$, given $t = g^\ell$ for some $\ell \in \mathbb{Z}$, calculate $\ell$. This is called the **discrete logarithm**, and is denoted $\log_g (t) = \ell$. 
A reminder of an earlier time...

Definition

\[ L_n(\alpha, c) = \exp \left( (c + o(1)) \left( \log n \right)^\alpha \left( \log \log n \right)^{1-\alpha} \right) \]
Last Time, on CSI: Discrete Logarithm

- The Discrete Logarithm Problem in groups with composite order can be decomposed.
- Solving Discrete Logarithm Problems is Hard.
- There are a set of algorithms that are deterministic
  - Brute Force runs in $O(n)$ and requires little storage.
  - Baby Step, Giant Step runs in $O(\sqrt{n})$ and requires $O(\sqrt{n})$ storage.
- There are more powerful algorithms that are probabilistic
  - Pollard’s $\rho$-method runs (heuristically, probabilistically) in $O(\sqrt{n})$ and requires little storage.
  - Index Calculus for problems in $\mathbb{F}_p$ runs (probabilistically) in $L_p\left(1/2, \sqrt{2}\right)$
Subsection 2

The Current State of Affairs
Algorithm Selection Depends on Setting

<table>
<thead>
<tr>
<th>$F_q$</th>
<th>Algorithm</th>
<th>Complexity</th>
<th>Field Size</th>
<th>Comp. Size</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_p$</td>
<td>NFS(^1)</td>
<td>$L_p [1/3]$</td>
<td>530-bit</td>
<td>8.65 CY</td>
<td>2007</td>
</tr>
<tr>
<td>$F_{2p}$</td>
<td>FFS(^2)</td>
<td>$L_q [1/3]$</td>
<td>613-bit</td>
<td>2.5 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$F_{p^n}$</td>
<td>FFS(^3)</td>
<td>$L_q [1/3]$</td>
<td>556-bit</td>
<td>0.001 CY</td>
<td>2005</td>
</tr>
</tbody>
</table>

\(^1\)[Joux-Lercer 2002]  
\(^2\)[Adleman 1994]  
\(^3\)[Joux 2006]
Algorithm Selection Depends on Setting

<table>
<thead>
<tr>
<th>$\mathbb{F}_q$</th>
<th>Algorithm</th>
<th>Complexity</th>
<th>Field Size</th>
<th>Comp. Size</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_p$</td>
<td>NFS</td>
<td>$L_p [1/3]$</td>
<td>530-bit</td>
<td>8.65 CY</td>
<td>2007</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^p}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>613-bit</td>
<td>2.5 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^n}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>556-bit</td>
<td>0.001 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^n}$</td>
<td>JIC$^4$</td>
<td>$L_q [1/3]$</td>
<td>1425-bit</td>
<td>0.06 CY</td>
<td>Jan 2013</td>
</tr>
</tbody>
</table>

$^4$[Joux, “Faster Index Calculus for the Medium Prime Case…”]
Algorithm Selection Depends on Setting

<table>
<thead>
<tr>
<th>$\mathbb{F}_q$</th>
<th>Algorithm</th>
<th>Complexity</th>
<th>Field Size</th>
<th>Comp. Size</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_p$</td>
<td>NFS</td>
<td>$L_p [1/3]$</td>
<td>530-bit</td>
<td>8.65 CY</td>
<td>2007</td>
</tr>
<tr>
<td>$\mathbb{F}_{2p}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>613-bit</td>
<td>2.5 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>FFS$^5$</td>
<td>$L_q [1/3]$</td>
<td>1971-bit</td>
<td>0.26 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>JIC2$^6$</td>
<td>$L_q [1/4 + o(1)]$</td>
<td>1178-bit</td>
<td>0.02 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^n}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>556-bit</td>
<td>0.001 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^n}$</td>
<td>JIC1</td>
<td>$L_q [1/3]$</td>
<td>1425-bit</td>
<td>0.06 CY</td>
<td>Jan 2013</td>
</tr>
</tbody>
</table>

$^5$[Göloğlu, Granger, McGuire, Zumbrägel]

$^6$[Joux, “A new index calculus algorithm...”]
Algorithm Selection Depends on Setting

<table>
<thead>
<tr>
<th>$F_q$</th>
<th>Algorithm</th>
<th>Complexity</th>
<th>Field Size</th>
<th>Comp. Size</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{2p}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>613-bit</td>
<td>2.5 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$F_{2^n}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>1971-bit</td>
<td>0.26 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$F_{2^n}$</td>
<td>JIC2</td>
<td>$L_q [1/4 + o(1)]$</td>
<td>1178-bit</td>
<td>0.02 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$F_{2^n}$</td>
<td>JIC2</td>
<td>$L_q [1/4 + o(1)]$</td>
<td>4080-bit</td>
<td>1.61 CY</td>
<td>Mar 2013</td>
</tr>
<tr>
<td>$F_{p^n}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>556-bit</td>
<td>0.001 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$F_{p^n}$</td>
<td>JIC1</td>
<td>$L_q [1/3]$</td>
<td>1425-bit</td>
<td>0.06 CY</td>
<td>Jan 2013</td>
</tr>
</tbody>
</table>
Algorithm Selection Depends on Setting

<table>
<thead>
<tr>
<th>$F_q$</th>
<th>Algorithm</th>
<th>Complexity</th>
<th>Field Size</th>
<th>Comp. Size</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{2p}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>613-bit</td>
<td>2.5 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$F_{2p}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>809-bit</td>
<td>4200 CY</td>
<td>Apr 2013</td>
</tr>
<tr>
<td>$F_{2n}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>1971-bit</td>
<td>0.26 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$F_{2n}$</td>
<td>JIC2</td>
<td>$L_q [1/4 + o(1)]$</td>
<td>1178-bit</td>
<td>0.02 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$F_{2n}$</td>
<td>JIC2</td>
<td>$L_q [1/4 + o(1)]$</td>
<td>4080-bit</td>
<td>1.61 CY</td>
<td>Mar 2013</td>
</tr>
<tr>
<td>$F_{2n}$</td>
<td>JIC2</td>
<td>$L_q [1/4 + o(1)]$</td>
<td>6120-bit</td>
<td>0.09 CY</td>
<td>Apr 2013</td>
</tr>
<tr>
<td>$F_{p^n}$</td>
<td>FFS</td>
<td>$L_q [1/3]$</td>
<td>556-bit</td>
<td>0.001 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$F_{p^n}$</td>
<td>JIC1</td>
<td>$L_q [1/3]$</td>
<td>1425-bit</td>
<td>0.06 CY</td>
<td>Jan 2013</td>
</tr>
</tbody>
</table>
Algorithm Selection Depends on Setting

<table>
<thead>
<tr>
<th>$\mathbb{F}_q$</th>
<th>Algorithm</th>
<th>Complexity</th>
<th>Field Size</th>
<th>Comp. Size</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_p$</td>
<td>NFS</td>
<td>$L_p \left[ \frac{1}{3} \right]$</td>
<td>530-bit</td>
<td>8.65 CY</td>
<td>2007</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^p}$</td>
<td>FFS</td>
<td>$L_q \left[ \frac{1}{3} \right]$</td>
<td>613-bit</td>
<td>2.5 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^p}$</td>
<td>FFS</td>
<td>$L_q \left[ \frac{1}{3} \right]$</td>
<td>809-bit</td>
<td>4200 CY</td>
<td>Apr 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>FFS</td>
<td>$L_q \left[ \frac{1}{3} \right]$</td>
<td>1971-bit</td>
<td>0.26 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>JIC2</td>
<td>$L_q \left[ \frac{1}{4} + o(1) \right]$</td>
<td>1178-bit</td>
<td>0.02 CY</td>
<td>Feb 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>JIC2</td>
<td>$L_q \left[ \frac{1}{4} + o(1) \right]$</td>
<td>4080-bit</td>
<td>1.61 CY</td>
<td>Mar 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>JIC2</td>
<td>$L_q \left[ \frac{1}{4} + o(1) \right]$</td>
<td>6120-bit</td>
<td>0.09 CY</td>
<td>Apr 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_{2^n}$</td>
<td>JIC2</td>
<td>$L_q \left[ \frac{1}{4} + o(1) \right]$</td>
<td>6168-bit</td>
<td>0.06 CY</td>
<td>May 2013</td>
</tr>
<tr>
<td>$\mathbb{F}_p$</td>
<td>FFS</td>
<td>$L_q \left[ \frac{1}{3} \right]$</td>
<td>556-bit</td>
<td>0.001 CY</td>
<td>2005</td>
</tr>
<tr>
<td>$\mathbb{F}_p$</td>
<td>JIC1</td>
<td>$L_q \left[ \frac{1}{3} \right]$</td>
<td>1425-bit</td>
<td>0.06 CY</td>
<td>Jan 2013</td>
</tr>
</tbody>
</table>
Modern Approaches to the Discrete Logarithm Problem

1 Introduction

2 Modern Approaches to the Discrete Logarithm Problem
   - The Number Field Sieve
   - The Function Field Sieve
   - Joux’s Index Calculus Algorithm 1: Pinpointing
   - Joux’s Index Calculus Algorithm 2: Relations from Perturbed Functions

3 Conclusion, Mk. II
Subsection 1

The Number Field Sieve
The Number Field Sieve is a descendent of the Quadratic Sieve:

- To factor a number $n$ find $x, y$ so that $x^2 \equiv y^2 \pmod{n}$ non-trivially.
- We then have that $\gcd(x - y, n)$ and $\gcd(x + y, n)$ are non-trivial factors of $n$.
- The way we generate such $x$ and $y$ is different than the Quadratic Sieve.
GNFS Factoring Process

- Preliminary Step I: Establish a Ring and Homomorphism and a Smoothness Base
  - We proceed by working over two rings, \( \mathbb{Z}/n\mathbb{Z} \) and a number field.
  - Select a smoothness bound \( B \) (over the number field, the bound is on the absolute norm of the element).
  - Our smoothness base is comprised of the \( k \) primes satisfying our smoothness bound.

- Run Collection Phase: Find Relations
  - Sieve on both sides, looking for relations (parity of the exponent of each term of the smoothness base expressed as elements of \( \mathbb{F}_2^k \)).

- Solve the resulting linear system
  - Once we have sufficient relations, we can use linear algebra to find elements that can be multiplied together to be squares.

- Post processing
  - Calculate square roots.
  - Map this square root to the integers via the ambient homomorphism.
  - Calculate factors using gcd.
Specializing for $\mathbb{F}_p$, $p > 5$:

- Let $l$ be an odd prime divisor of $p - 1$
  - Note: we must be able to factor $p - 1$.

- Let $B$ be our smoothness bound and treat $a \in \mathbb{F}_p$ as $B$-smooth if $a \in \mathbb{Z}$ is $B$-smooth.

- Let $g \in \mathbb{F}_p^\times$ and $t \in \langle g \rangle$, both $B$-smooth.
  - We have already seen how to proceed if $t$ is not $B$-smooth.

- We choose $R_1$ and $R_2$ as either:
  - A number field and the integers (for logarithms in $\mathbb{F}_p$).
  - Two number fields (in this case, we still call the algorithm “The Number Field Sieve”).
  - Two function fields (in this case, we then call the algorithm “The Function Field Sieve”)

NFS for Logarithms: Preliminaries I
We need the rings $R_i$ to come with (easy to find) homomorphisms from $\phi_i : R_i \to \mathbb{F}_p$.

We construct $l$th powers, $\alpha_i \in R_i$ so that $\phi_1(\alpha_1) = \phi_2(\alpha_2)$.

We then have $\ell_l \equiv -\log_g t \pmod{l}$.

Once we know $\ell_l$ for all values $l$ dividing $p - 1$, we can calculate $\ell$ via the CRT.
Choose a parameter \( d \) so that \( \log_2 p > d \geq 1 \).

Take \( m = \left\lfloor d\sqrt{p} \right\rfloor \) and construct the base-\( m \) representation of \( p \) with \( m \)-ary digits \( a_i \):

\[ p = \sum_{i=0}^{d} a_i m^i \]

Take \( f(x) = \sum_{i=0}^{d} a_i x^i \). This is irreducible over \( \mathbb{Q} \).

Let \( \alpha \) denote a root of \( f \) in \( \overline{\mathbb{Q}} \) and take \( R_1 = \mathbb{Z} \) and \( R_2 = \mathbb{Z}[\alpha] \).

The map \( \phi_1 \) is just reduction mod \( p \).

The map \( \phi_2 \) is the map sending the monomial \( b_i \alpha^i \mapsto b_i m^i \) (mod \( p \)), respecting addition.
Sieving

- This proceeds in the same way as with the GNFS (factoring) algorithm.
- Produces factorizations of the sieved $B$-smooth elements in our respective rings.
- Operates elements of the form $(a - bm) \in R_1$ and $(a - b\alpha) \in R_2$.
- Heuristic performance: $L_p(1/3)$ pairs must be tested.
This seems like a job for... Gaussian Elimination!

Sadly our old friend is $O(r^3)$, which would ruin our bound.

We use some combination of the Block Wiedmann algorithm, the Lanczos algorithm, and structured Gaussian Elimination.

Results in (probable) $l$th powers, which we use to solve the logarithm.
Due to [Kleinjung, 2007]:

- $p$ was chosen as a 532-bit prime so that $(p - 1)/2$ is prime (based on a scaled value of $\pi$).
- $g = 2$ is chosen (and generates $\mathbb{F}_p^\times$).
- An arbitrary target $t$ is chosen (based on a scaled value of $e$).
NFS Computing Time

- 831266637 relations generated in 6.6 core-years.
- Duplicates are discarded, resulting in 423671492 relations.
- Removing singletons and cliques, gives us a $2177226 \times 2177026$ matrix with 289976350 non-zero entries.
- Processing via the Block Wiedemann algorithm in about 28 core-years.
- Post processing was accomplished in a few hours.
Subsection 2

The Function Field Sieve
Working in $\mathbb{F}_{q^n}$ with $q = p^k$.

As a field, this is obviously $\mathbb{F}_{p^k}$, but it suits us to tune the extension degree.

Our smoothness bases are ideals whose norms are polynomials of small degree.

In certain cases (when $\log q$ and $\sqrt{n} \log n$ have the right relation) our smoothness bases are ideals whose norms are degree 1 polynomials.
Choose parameters $d_1, d_2$ minimally so that $d_2 = d_1$ or $d_2 = d_1 + 1$, and $d_1 d_2 > n$.

Choose $g_1(x)$ of degree $d_1$ and $g_2(x)$ of degree $d_2$ in $\mathbb{F}_q[x]$ so that $g_2(g_1(T)) + T$ has an irreducible degree $n$ factor over $\mathbb{F}_q$, $F(T)$.

We then have

$$\mathbb{F}_{q^n} \cong \mathbb{F}_q [T] / \langle F(T) \rangle$$

Define

$$f_1(X, T) = X - g_1(T) \text{ and } f_2(X, T) = g_2(X) + T$$

$f_1$ and $f_2$ have a common root $X = g_1(T)$.

We use these polynomials to define our function fields.
We examine elements of the form $a(T)X - b(T)$ in the two function fields.

We’ll consider only $a(T) = wT + 1$ and $b(T) = uT + v$ where $w, u, v \in \mathbb{F}_q$.

Compute the norm of these elements in the two function fields, keeping elements whose norms are smooth (i.e., whose norms are linear polynomials).

We heuristically assume that these elements “act” like random independent polynomials in the two function fields.

Due to our choice of $f_1$ and $f_2$ our smooth elements are a very special form; there is a $u \in \mathbb{F}_q$ so that:
- our smooth elements on the linear side are of the form $T + u$.
- our smooth elements on the non-linear side are of the form $T + g(u)$.

If we assume that this process produces random looking polynomials this occurs with probability better than $1/((d_1 + 1)! \cdot (d_2 + 1)!)$.

Sieving occurs in $L_{q^n}(1/3)$. 
Our relations can be transformed into linear equations involving:
- logarithms of polynomials on the linear side.
- logarithms of (principal) ideals on the other side.

The actual linear algebra occurs as before.
And then...

- Sadly, logarithms of degree 1 polynomials aren’t sufficient.
- “Large” elements must be presented as a product of these linear terms.
- This uses a technique called “special-q descent”.
  - We want the logarithm of $y$.
  - Build $y^i T^j$ until we find an element that can be factored into polynomials of degree $< \mu \sqrt{n}$. Let $q$ be one such polynomial. ($\mu$ is a parameter chosen so that $\mu \in (1/2, 1)$).
  - Sieve polynomials of the form $a(T)X - b(T)$ where $\deg a(T), \deg b(T) \leq \mu \sqrt{n}$.
  - Look for elements whose factors are of degree strictly smaller that $\deg q$.
  - Wash, rinse, repeat.
  - Backtrack once we have descended to degree 1.
Due to [Joux, 2006]:

- $p = 370801$, our field is $\mathbb{F}_{p^{30}}$, a 556-bit cardinality, with multiplicative group of cardinality 114 bits.
- Here $q = p$. 
FFS Computing Time

- 329082 relations generated in 45 core-minutes.
- Removing singletons and cliques gives us 150270 equations in 148270 unknowns.
- Processing via the Lanczos algorithm in about 80 core-hours.
- Special-\(q\) descent took 40 core-minutes.
Subsection 3

Joux’s Index Calculus Algorithm 1: Pinpointing
Any level of processing on bad candidates is wasted time.

From a complexity view for the sieving stage, we care not just about the number of successful candidates, but the total number tested.

For fields with a “medium size” subfield we can use “pinpointing”.

We otherwise use the prior FFS algorithm.
Construct $X = Y^{d_1}$ and $Y = g_2(X)$, where $g_2$ is degree $d_2$.

After normalization, we consider candidates of a certain form, where $a, b, c \in \mathbb{F}_q$:

$$Y^{d_1+1} + aY^{d_1} + bY + c = Xg_2(X) + aX + bg_2(X) + c$$

This yields a relation when both sides factor into linear polynomials.
One-Sided Pinpointing

- Look for polynomials of a form that will split on the left hand side by picking $B, C \in \mathbb{F}_q$:

$$U^{d_1+1} + U^{d_1} + BU + C$$

- This will require approximately $(d_1 + 1)!$ candidates.

- Once one is found, we can amplify it by performing the change of variable $U = Y/a$, with $a \in \mathbb{F}_q^\times$.

- The amortized cost of these relations is much better than the cost of sieving.
A similar procedure over some fields (e.g., Kummer Extensions) allows us to perform similar tricks on both sides. This decreases the amortized cost even more. The 1425-bit discrete logarithm problem mentioned previously uses this approach.
Subsection 4

Joux’s Index Calculus Algorithm 2: Relations from Perturbed Functions
Formal Place Setting (Look! A Shrimp Fork!)

- Specified as applying to fields of the form $\mathbb{F}_{q^{2k}}$ where $q \approx k$.
- The characteristic of $\mathbb{F}_{q^{2k}}$ is required to be very small (ideally fixed!)
- In pinpointing we amplified a single equation to a class of equations through a linear change of variables.
- This approach notes that if we broaden our transforms, we can rely on a single polynomial for all relations.
Who is that masked man?

- We transform using a Möbius transform!

\[ x \mapsto \frac{ax + b}{cx + d} \]

- Multiply by the denominator in the appropriate degree to get a polynomial.

\[ f(x) \mapsto F_{a,b,c,d}(x) = (cx + d)^{\text{deg} f} f \left( \frac{ax + b}{cx + d} \right) \]

- In the case that \( f \) splits into monic irreducible factors, it induces a factorization of \( F_{a,b,c,d} \):

\[
f(Y) = \prod_{i=1}^{k} F_i(Y)^{e_i} \mapsto F_{a,b,c,d}(x) \prod_{i=1}^{k} \left( (cx + d)^{\text{deg} F_i} F_i \left( \frac{ax + b}{cx + d} \right) \right)^{e_i}
\]
Every single $f$ produces $q^3$ candidates.

What should we choose for $f$?

$f(x) = x^q - x$ splits by design.
Consider $\mathbb{F}_{q^{2k}}$ as an extension over $\mathbb{F}_{q^2}$, where $k \leq q + \delta$ ($\delta$ small compared to $q$).

- Take $h_0(X), h_1(X) \in \mathbb{F}_{q^2}[X]$ so that $h_1(X)X^q - h_0(X)$ has an irreducible factor $l(X)$ of degree $k$.

- It is (heuristically) likely that we can find linear $h_i(X)$ that satisfy this requirement.

- We then view $\mathbb{F}_{q^{2k}} \cong \mathbb{F}_{q^2}[X]/(l(X))$. 
Start with:

\[
\prod_{\alpha \in \mathbb{F}_q} (Y - \alpha) = Y^q - Y.
\]

Apply the above change of variable to \( Y \) (with \( a, b, c, d \in \mathbb{F}_q^2 \) and \( ad - bc \neq 0 \))

\[
(cX + d) \prod_{\alpha \in \mathbb{F}_q} ((a - \alpha c)X + (b - \alpha d))
\]

\[
= (cX + d)(aX + b)^q - (aX + b)(cX + d)^q
\]
It gets better!

- Evaluate (2).

\[
\frac{(ca^q - ac^q)Xh_0(X) + \cdots + (db^q - bd^q)h_1(X)}{h_1(X)} \quad \text{(mod } l(X))
\]

- If we add \( h_1(X) \) to our smoothness base, we get a relation whenever the numerator splits into linear factors.

- These will not all be distinct (indeed, this is why we operate over \( \mathbb{F}_{q^2} \) and not \( \mathbb{F}_q \).

- We expect to find enough relations after \( O(p^2) \) quadruples.

- Linear algebra then gives us the logs of the degree one terms.
Evaluate (2).

\[
\frac{(ca^q - ac^q)Xh_0(X) + \cdots + (db^q - bd^q)h_1(X)}{h_1(X)} \pmod{l(X)}
\]

If we add \(h_1(X)\) to our smoothness base, we get a relation whenever the numerator splits into linear factors.

These will not all be distinct (indeed, this is why we operate over \(\mathbb{F}_{q^2}\) and not \(\mathbb{F}_q\).

We expect to find enough relations after \(O(p^2)\) quadruples.

Linear algebra then gives us the logs of the degree one terms.

In essentially polynomial time…
We really can’t just proceed with logs of degree 2 terms, but we needn’t calculate all such logs.

Lazy evaluation in the descent stage has produced the best performance.
First, spend some time calculating $g^i t$ until it decomposes into “reasonably low” degree.

With fixed characteristic fields, we can fix one of the coefficients used in special-$q$ descent.

We use this “classical” special-$q$ descent for early descent, and then pass to a new descent algorithm.
Given a polynomial $Q$ of degree $D$ find pairs of polynomials, $k_1, k_2$ of degree $d = \lceil (D + 1) \rceil 2$ so that $Q(X)$ divides $k_1(x)^q k_2(x) - k_1(x) k_2(x)^q$ (mod $l(X)$).

With good probability, we obtain a relation between $Q$ and polynomials of at most $d$.

This is a bilinear system!

We can search for such $k_1, k_2$ using a Gröbner basis algorithm.

If $D$ is “large”, we should instead find $k_1$ of degree $d$ and $k_2$ of degree $D + 1 − d$. 
The Final Algorithm

- Use the special $q$-descent to degree $\sqrt{q}$ and then the new descent algorithm after that.
- This results in complexity $L(1/4 + o(1))$. 
Section 3

Conclusion, Mk. II
“The understanding of the hardness of the DLP in the multiplicative group of finite extension fields could be said to be undergoing a mini-revolution.” — From GGMZ “Solving a 6120-bit DLP on a Desktop Computer”

- Solving Discrete Logarithm Problems is Hard.
Today’s Conclusion

“The understanding of the hardness of the DLP in the multiplicative group of finite extension fields could be said to be undergoing a mini-revolution.” — From GGMZ “Solving a 6120-bit DLP on a Desktop Computer”

- Solving Discrete Logarithm Problems is Hard.
- But not as hard as it used to be in some settings...
Thank You!
The principal font is Evert Bloemsma’s 2004 humanist san-serif font Legato. This font is designed to be exquisitely readable, and is a significant departure from the highly geometric forms that dominate most san-serif fonts. Legato was Evert Bloemsma’s final font prior to his untimely death at the age of 46.

Math symbols from the MathTime Professional II (MTPro2) fonts, a font package released in 2006 by the great mathematical expositor Michael Spivak.

The URLs are typeset in Luc(as) de Groot’s 2005 Consolas, a monospace font with excellent readability.