On the discrete logarithm problem in finite fields

Pierrick Gaudry

CNRS, Université de Lorraine, Inria
Nancy, France

joint work with Razvan Barbulescu, Antoine Joux, Emmanuel Thomé

RICAM – Linz, Austria

Plan

Background

Recent history in small / medium characteristic

Quasi-polynomial in small characteristic

Discussion about the heuristics

The Discrete Log Problem

Definition: the discrete log problem

Let G be a cyclic group of order N, with a generator g.
The DLP is:

$$\text{Given } h \in G, \text{ find an integer } x \text{ such that } h = g^x.$$

Classical assumptions:

- The order N is known (usually, also its factorization).
- The group G is effective, i.e. we have
  - a compact representation of the elements of G (ideally, in $O(\log N)$ bits);
  - an efficient algorithm for the group law (polynomial time in $\log N$).

Rem: the integer x makes sense only modulo N.

The Pohlig-Hellman reduction

Let $N = \prod p_i^{e_i}$ be the factorization of the group order.
Let $g_i = g^{N/p_i^{e_i}}$ and $h_i = h^{N/p_i^{e_i}}$.
Then, g_i is of order $p_i^{e_i}$ and

$$h_i = g_i^{x_i}, \quad \text{where} \quad x_i \equiv x \mod p_i^{e_i}.$$

Thm. Using the Chinese Remainder Theorem, the DLP in G reduces to DLPs in groups whose orders are prime powers.

A similar trick, *à la* Hensel, allows to reduce the DLP modulo a prime power to several DLPs modulo primes.

Theorem (Pohlig-Hellman reduction)

The DLP in G cyclic of composite order is not harder than the DLP in the subgroup of G of largest prime order.

Shanks' baby-step giant-step algorithm

Let K be a parameter (in the end, $K \approx \sqrt{N}$). Write the dlog x as

$$x = x_0 + K\,x_1, \quad \text{with} \quad 0 \leq x_0 < K \text{ and } 0 \leq x_1 < N/K.$$

Algorithm:

1. Compute **Baby Steps**:
   For all i in $[0, K-1]$, compte g^i.
   Store in a hash table the resulting pairs (g^i, i).
2. Compute **Giant Steps**:
   For all j in $[0, \lfloor N/K \rfloor]$, compute hg^{-Kj}.
   If the resulting element is in the BS table, then get the corresponding i, and return $x = i + Kj$.

Theorem

Discrete logarithms in a cyclic group of order N can be computed in less than $2\lceil\sqrt{N}\rceil$ operations.

Summary of generic algorithms

Putting things together, one obtain:

Theorem (DLP in generic groups)

Let G be a cyclic group of order N, and let p be the largest prime factor of N. The DLP in G can be solved in $O(\sqrt{p})$ operations in G (up to factors that are polynomial in $\log N$).

Thm. This is optimal (work of Nechaev, Shoup).

Rem. The BSGS algorithm has a large space $O(\sqrt{p})$ complexity. Variants of Pollard's Rho method provide a low-memory, easy to parallelize alternative to be used in practice (but heuristic).

Finite fields are **not** generic groups!

Smoothness (CEP and PGF)

Def. An integer (resp. a polynomial over $\mathbb{F}_q$) is B-**smooth** if all its prime factors are $\leq B$ (resp. all irred. factors have deg $\leq B$).
Thm. The proportion of y-smooth integers less than x (resp. of m-smooth polynomials of degree less than n) is

$$u^{-u(1+o(1))},$$

where $u = \log x / \log y$ (resp. $u = n/m$). *[+ additional conditions]*

Usually restated with the L-**notation**: for $\alpha \in [0,1]$ and $c > 0$, define

$$L_N(\alpha, c) = \exp\left(c(\log N)^{\alpha}(\log\log N)^{1-\alpha}\right).$$

An integer less than $L_N(\alpha)$ is $L_N(\beta)$-smooth with probability $L_N(\alpha - \beta)^{-1+o(1)}$.

$L(1/2)$ index calculus in $\mathbb{F}_{2^n} = \mathbb{F}_2[x]/\varphi(x)$

Algorithm: To compute the log of h in base g:

0. Fix a smoothness bound B, and construct the factor base $\mathcal{F} = \{p_i \text{ irreducible}; \deg p_i \leq B\}$.
1. **Collect relations**. Repeat the following until enough relations have been found:
   1.1 Pick a at random and compute $z = g^a$.
   1.2 Seen as a poly of degree $< n$, check if z is smooth.
   1.3 If yes, write z as a product of elements of $\mathcal{F}$ and store the corresponding relation as a row of a matrix.
2. **Linear algebra**. Find a vector v in the right-kernel of the matrix, modulo $2^n - 1$. Normalizing to get $\log g = 1$, this gives the log of all factor base elements.
3. **Individual logs**. Pick b at random until h^b is smooth. Deduce the log of h.

$L(1/2)$ index calculus: comments

Choosing $B = \log_2 L_{2^n}(\frac{1}{2}, \sqrt{2}/2)$, we get a **total running time** of

$$L_{2^n}\Big(\frac{1}{2}, \sqrt{2} + o(1)\Big).$$

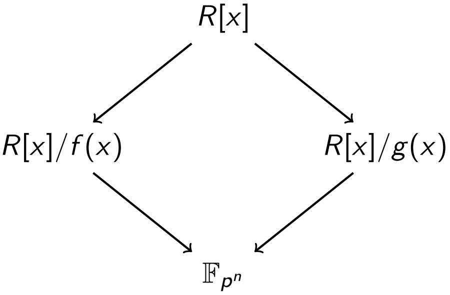
Rem. All $L(1/2)$ and $L(1/3)$ DLP algorithms (i.e. all known algorithms before 2013) follow the same scheme:

- Relation collection;
- Linear algebra to get log of factor base elements;
- Individual log, to handle any element.

Joux's $L(1/4)$ algorithm of 2013 still uses this terminology (but very different in nature).
Quasi-polynomial time algorithm: it's time to stop speaking about factor base!

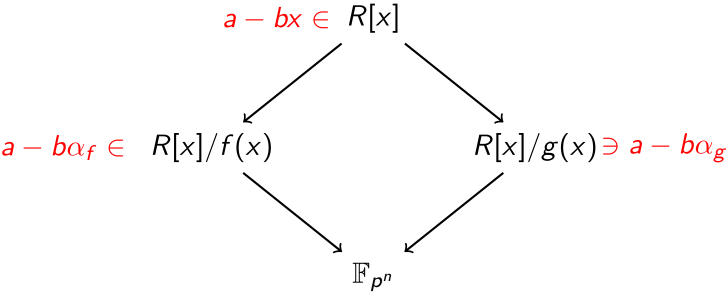
The key to $L(1/3)$ algorithms

Find a ring R, and monic polynomials $f(x)$ and $g(x)$ over R such that we have a **commutative diagram** as follows:



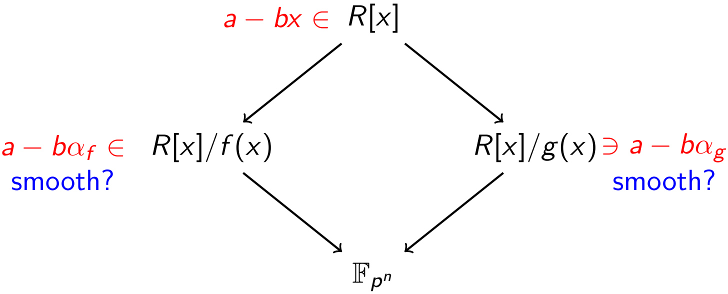
The key to $L(1/3)$ algorithms

Find a ring R, and monic polynomials $f(x)$ and $g(x)$ over R such that we have a **commutative diagram** as follows:



The key to $L(1/3)$ algorithms

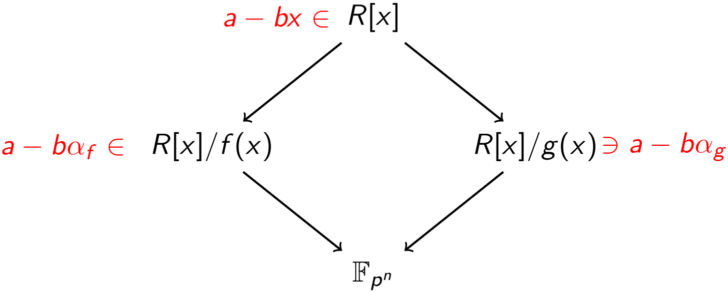
Find a ring R, and monic polynomials $f(x)$ and $g(x)$ over R such that we have a **commutative diagram** as follows:



If smooth on both sides, then we get a **relation** in $\mathbb{F}_{p^n}$.

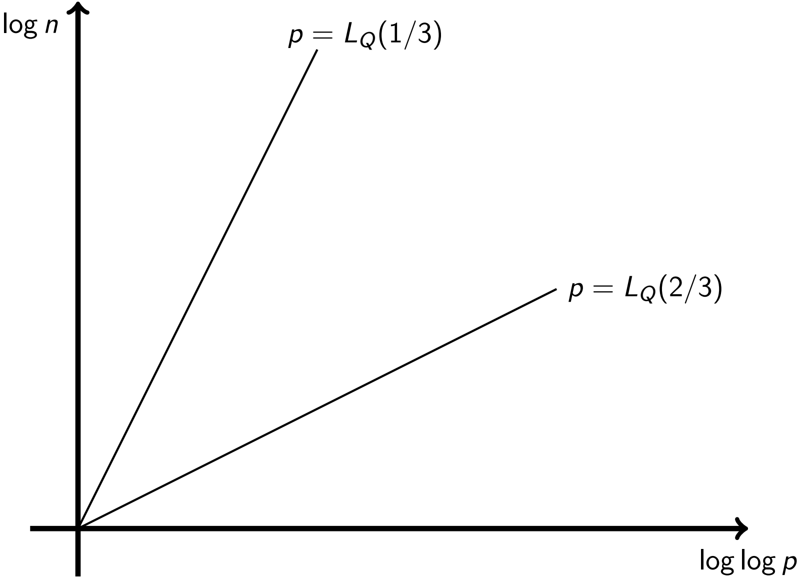
Make sure the elements $a - b\alpha_f$ and $a - b\alpha_g$ are **small**: $L_{p^n}(2/3)$.

The key to $L(1/3)$ algorithms

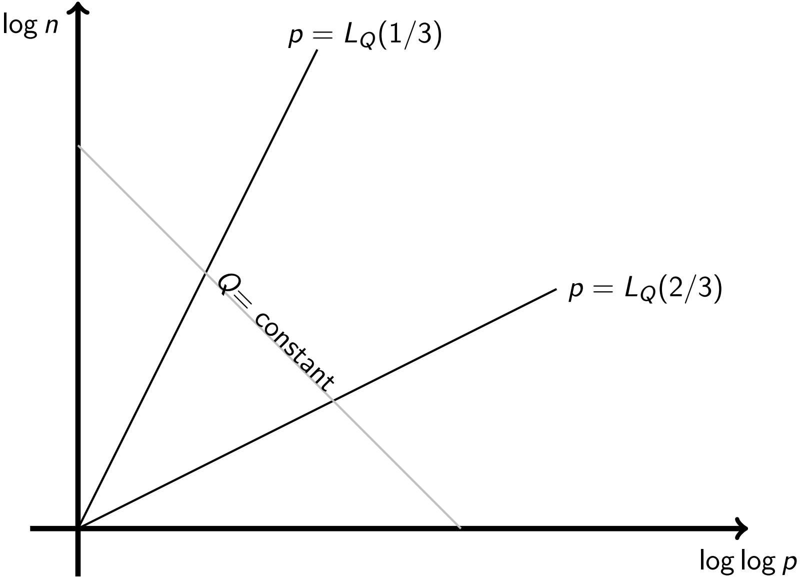


NFS (Number Field Sieve): $R = \mathbb{Z}$. Many ways to choose f and g depending on the sizes of p and n. *works for large p*
FFS (Function field Sieve): $R = \mathbb{F}_p[t]$. Less variants for choosing f and g. *works for large n*

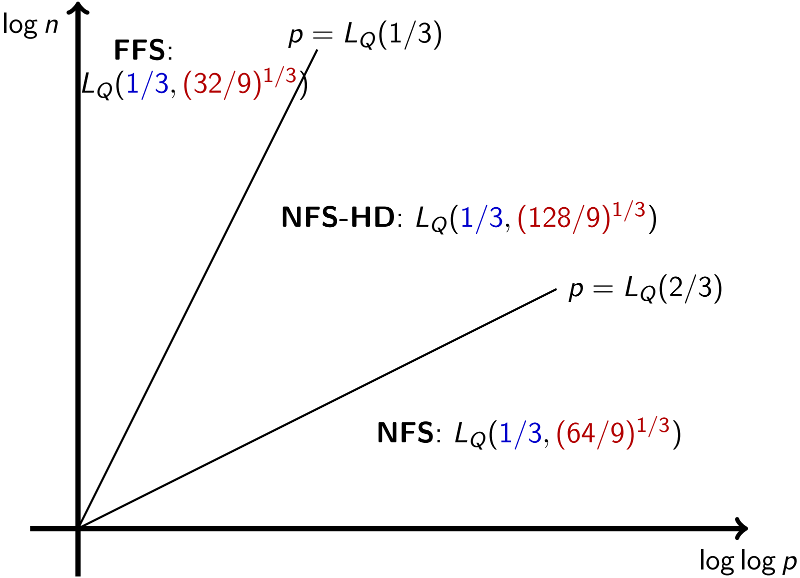
DL complexity in $\mathbb{F}_Q$ with $Q = p^n$



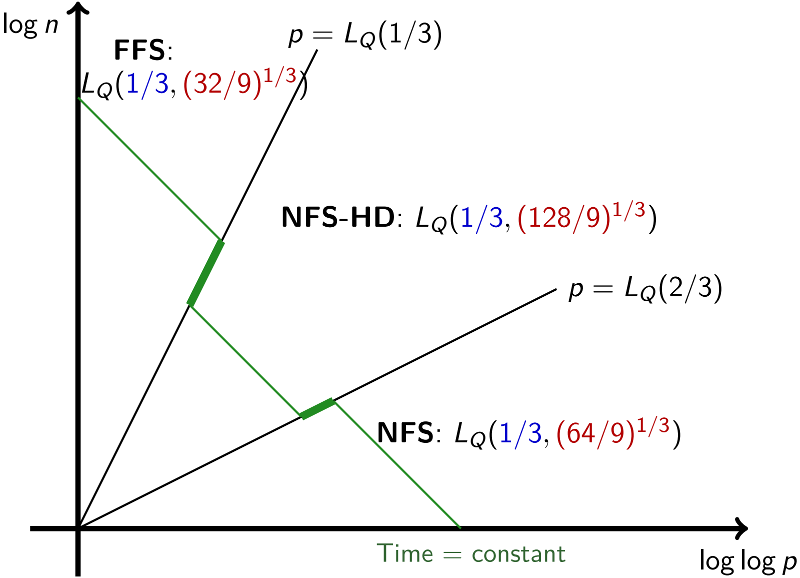
DL complexity in $\mathbb{F}_Q$ with $Q = p^n$



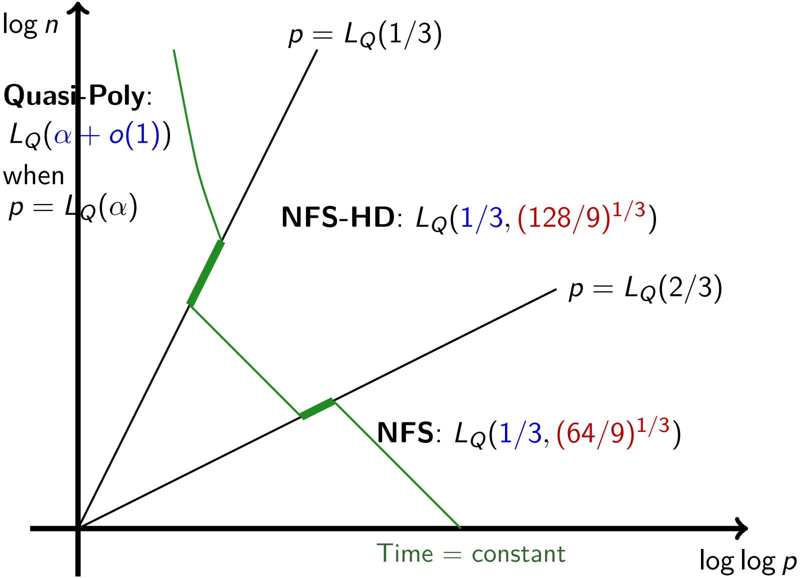
DL complexity in $\mathbb{F}_Q$ with $Q = p^n$



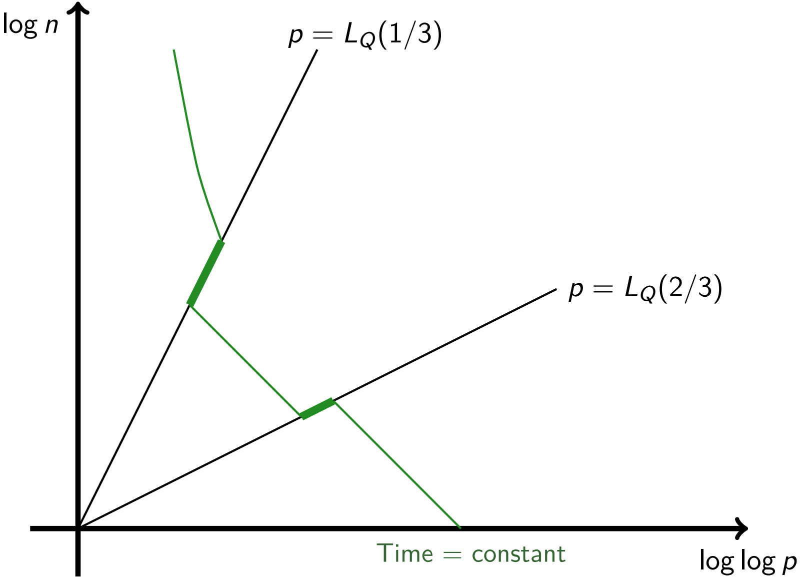
DL complexity in $\mathbb{F}_Q$ with $Q = p^n$



DL complexity in $\mathbb{F}_Q$ with $Q = p^n$



DL complexity in $\mathbb{F}_Q$ with $Q = p^n$



Plan

Background

Recent history in small / medium characteristic

Quasi-polynomial in small characteristic

Discussion about the heuristics

Preliminary results

In 2012, Hayashi-Shimoyama-Shinohara-Takagi computed discrete logs in $\mathbb{F}_{3^{6 \cdot 97}}$.

Algorithm: FFS, but the medium-sized subfield played a key role to speed-up the computation.

From lower-medium prime to small characteristic

End of 2012 – beginning of 2013: the **pinpointing** trick.

- Invented by Joux;
- Much faster relation collection;
- Initially for FFS in the medium prime range;
- Works in small characteristic for composite extension;
- New records: $\mathbb{F}_{33341353^{57}}$ and $\mathbb{F}_{2^{1778}}$.

Beginning of 2013: **other ideas** in the same spirit.

- Invented by Göloğlu-Granger-McGuire-Zumbrägel;
- Polynomial-time algorithm for logarithms of linear polynomials;
- Complexity in the best case: $L_{q^n}(1/3, 2/3)$;
- New record: $\mathbb{F}_{2^{1971}}$.

The $L(1/4)$ algorithm of Joux

New **features** of the $L(1/4 + o(1))$ algorithm:

- The "factor base" is reduced to polynomials of degree 1 and 2.
- The complexity is given solely by the individual logarithm phase.
- The descent for individual logarithms is split in two steps:
  - A classical FFS-like descent;
  - A brand-new descent using polynomial systems, in a variant due to Pierre-Jean Spaenlehauer.
- Joux remarks that if we could solve polynomial systems in polynomial time (!) this would give a quasi-polynomial algorithm for the DLP.

Amazing record computations

During Spring 2013, **big competition** between Joux and the Irish team.

- 22 Mar 2013, Joux: $\mathbb{F}_{2^{4080}}$.
- 11 Apr 2013, Göloğlu et al.: $\mathbb{F}_{2^{6120}}$.
- 21 May 2013, Joux: $\mathbb{F}_{2^{6168}}$.

Rem. Kummer extensions play a crucial role.

Plan

Main result

Main result (based on heuristics)

Let K be a finite field of the form $\mathbb{F}_{q^k}$. A discrete logarithm in K can be computed in heuristic time

$$\max(q, k)^{O(\log k)}.$$

Applications of the main result

The result holds for any field, but is interesting for small to medium characteristic:

- **Very small characteristic**: $K = \mathbb{F}_{2^n}$, with prime n. Complexity is $n^{O(\log n)} = 2^{O((\log n)^2)}$. Much better than $L_{2^n}(1/3) \approx 2^{\sqrt[3]{n}}$.
- **Characteristic is polynomial in Q**: $K = \mathbb{F}_{q^k}$, with $q \approx k$. Complexity is $\log Q^{O(\log \log Q)}$, where $Q = \#K$. Again, this is $L_Q(o(1))$.
- **Characteristic is sub-exponential in Q**: $K = \mathbb{F}_{q^k}$, with $q \approx L_{q^k}(\alpha)$. Complexity is $L_{q^k}(\alpha + o(1))$, i.e. better than Joux-Lercier or FFS for $\alpha < 1/3$.

Setting

The setting of the algorithm is the same as for Joux's $L(1/4)$ algorithm:

$K = \mathbb{F}_{q^{2k}}$, with $k \approx q$.
The field $\mathbb{F}_{q^2}$ is represented in any usual way.

The **extension** of degree k is constructed as follows:

- Take h_0 and h_1 two polynomials over $\mathbb{F}_{q^2}$, of small degree (2 should be ok, but heuristic).
- Let $\Phi(X) = h_1(X)X^q - h_0(X)$.
- Until there is an irreducible factor $I(X)$ of $\Phi(X)$ of degree k.

Rem. This works only if $k \leq q + 2$.

How to fit in this setting?

If the given field $\mathbb{F}_{p^n}$ is such that $n > p + 2$, we **embed** the DL in $\mathbb{F}_{p^n}$ into a **larger field**:
Let q be the smallest power of p such that $q + 2 \geq n$ and set $k = n$.

Then, $\mathbb{F}_{q^{2k}}$ contains $\mathbb{F}_{p^n}$ and we are in the previous setting.
The cost of this embedding is reflected by the max() in the formula of the complexity.

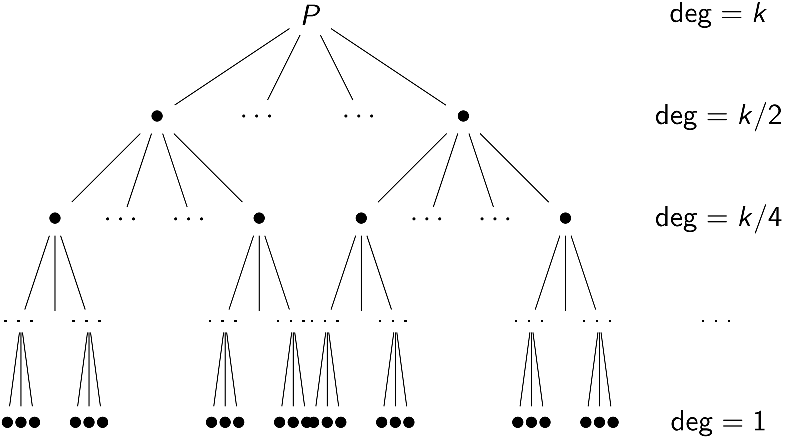
Rem. If n is composite, it might not be necessary to pay as much for this extension.

General strategy

Given an element $P(x)$ in $\mathbb{F}_{q^{2k}}$ represented as a polynomial of degree $D \leq k-1$ over $\mathbb{F}_{q^2}$, we are going to **descend** it:

- Find a linear relation between $\log P$ and the logs of elements of degrees at most $D/2$;
- Do it **recursively**: each new log can be again expressed in terms of logs of polynomials of smaller degrees;
- Go down to degree 1;
- The logs of all linear polynomials can be found in polynomial-time in q. (Already known from Göloğlu et al.)

Descent tree



One step of descent

Proposition (heuristic)

Let $P(X) \in \mathbb{F}_{q^2}$ of degree $D < k$. In time polynomial in D and q, we find an expression

$$\log P = e_1 \log P_1 + \cdots + e_m \log P_m,$$

where $\deg P_i \leq D/2$, and the number m of P_i is in $O(q^2 D)$.

Provided that the logs of linear polynomials can be computed in polynomial time in q, then the main result follows from the analysis of the size of the descent tree.

The descent tree

Each node of the descent tree corresponds to one application of the Proposition, hence its arity is in $q^2 D$.

level	$\deg P_i$	width of tree
0	k	1
1	$k/2$	$q^2 k$
2	$k/4$	$q^2 k \cdot q^2 \frac{k}{2}$
3	$k/8$	$q^2 k \cdot q^2 \frac{k}{2} \cdot q^2 \frac{k}{4}$
$\vdots$	$\vdots$	$\vdots$
$\log k$	1	$\leq q^{2 \log k} k^{\log k}$

Total number of nodes $= q^{O(\log k)}$.
Each node yields a cost that is polynomial in q, hence the result.

One step of descent: how?

Start from the field equation:

$$X^q - X = \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} (\beta X - \alpha),$$

Plug the input $P(X)$, twisted by an homography $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$:

$$\begin{aligned}(aP(X)+b)^q(cP(X)+d) - (aP(X)+b)(cP(X)+d)^q \\ = \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} \beta(aP(X)+b) - \alpha(cP(X)+d) \\ = \lambda \prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} P(X) - m^{-1}\cdot(\alpha:\beta).\end{aligned}$$

One step of descent: how?

Left-hand side:
Let the q-power come inside the formulae, and use $X^q \equiv h_0(X)/h_1(X)$.
For instance,

$$(aP(X) + b)^q = a^q \tilde{P}(X^q) + b^q \equiv a^q \tilde{P}(\frac{h_0}{h_1}) + b^q.$$

Hence, modulo denominator clearing, it is a polynomial of degree $O(\deg P)$.
Probability that LHS splits in polys of degree $\leq \frac{1}{2} \deg P$ is constant.

Right-hand side:
All factors are in $\left\{ P(X) - \gamma \mid \gamma \in \mathbb{F}_{q^2} \right\}$.

One step of descent: how?

Now, we let the matrix $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ vary.

The RHS is the same as for $m = \mathrm{Id}$ if m is in $PGL_2(\mathbb{F}_q)$.
The appropriate set where to pick m is the set of cosets:

$$\mathcal{P}_q = PGL_2(\mathbb{F}_{q^2})/PGL_2(\mathbb{F}_q).$$

For any $\mathfrak{q}$, the order of $PGL_2(\mathbb{F}_\mathfrak{q}) = \mathfrak{q}^3 - \mathfrak{q}$, so

$$\#\mathcal{P}_q = q^3 + q.$$

Conclusion: Have $\Theta(q^3)$ relations; need q^2 to eliminate the right-hand sides. More than enough! (but heuristic)

Running-time estimates

Loop for each representative m of $\mathcal{P}_q$: $O(q^3)$ elements.
For each m, we have to

- Write the corresponding LHS of degree $O(k)$.
- Test its smoothness.
- If it is smooth, write the corresponding RHS.

Fact 1: the linear system is constructed in polynomial time.

It has $\Theta(q^3)$ rows and $O(q^2)$ columns.
Fact 2: the linear system is solved in polynomial time.

The system has $O(q)$ non-zero entries per rows: rather sparse.

Logarithms of linear polynomials

Strategy: set $P(X) = X$ in the same machinery as before.

All LHS have the same as degrees as h_0 and h_1, say 2.
The probability that they split into linear factors is $1/2$.

By construction, the RHS is a product of linear factors.

Conclusion: Have $\Theta(q^3)$ relations; expect to need $O(q^2)$ to get a full rank matrix. Again, more than enough! (but heuristic)

Rem: Here, this is a kernel computation, whereas inside the descent tree, we solve inhomogenous systems.

The result

Main result (based on heuristics)

Let K be a finite field of the form $\mathbb{F}_{q^k}$. A discrete logarithm in K can be computed in heuristic time

$$\max(q, k)^{O(\log k)}.$$

Plan

Background

Recent history in small / medium characteristic

Quasi-polynomial in small characteristic

Discussion about the heuristics

Summary of heuristics

The sucess of the algorithm relies on three main heuristics:

- One can find appropriate h_0 and h_1 of low degree to define the extension.
- When descending a P, at each node, we get enough relations, and the corresponding system is solvable.
  - Smoothness probability.
  - Full rank question.
- The linear system corresponding to linear polynomials is full-rank.
  Very similar to the previous one, but slightly different.

Resilience of the algorithm

When trying to prove smoothness or rank results, one can allow partial results:

- **Random self-reducibility** of the DLP.
  If an algorithm can compute the logs of a non-trivial fraction of the elements, then one can compute the logs of all of them.
  [multiply by a random power of the generator]
- **Re-randomization inside the algorithm.**
  At a node of the tree, we usually have a lot of choices.
  If some child is problematic, choose another relation not involving that one.

So, why aren't we happy with heuristics?

Despite numerous experiments, we didn't realize that as stated our heuristics could not hold:

- Paper by **Cheng, Wan and Zhuang**.
  If $P(x)$ divides $h_1(X)X^q - h_0(X)$, then its log can not be found with our strategy.
- Paper by **Huang and Narayanan** (last week!)
  Problems when there is a large ℓ such that ℓ^2 divides some multiplicative group.

In both cases, the authors also show how to fix the problem if it occurs.

The heuristic on h_0 and h_1

Problem: Given q and $k \leq q+2$, find h_0 and h_1 of degree 2 in $\mathbb{F}_{q^2}[X]$ such that $h_1(X)X^q - h_0(X)$ has an irreducible factor of degree k.

- As such, looks very hard. Although somehow easier than the similar heuristic for "polynomial selection" in FFS.
- It is possible to allow the degree of h_0 and h_1 to be larger than 2.

  Any constant gives the same complexity, and maybe allowing something that grows slowly to inifinity is acceptable.
- Also, it is possible to use q^2, q^3, or any constant power of q instead of q.

  It corresponds to embedding the problem in a larger field: the change in the overall complexity can stay under control.

The smoothness heurstic

Problem: Given q, h_0, h_1 and a polynomial $P(X)$, what proportion of a, b, c, d in $\mathbb{F}_{q^2}$ yield a $\frac{1}{2}\deg P$-smooth polynomial:

$$(a^q\tilde{P}(\frac{h_0}{h_1}) + b^q)(cP(\frac{h_0}{h_1}) + d) - (aP(\frac{h_0}{h_1}) + b)(c^q\tilde{P}(\frac{h_0}{h_1}) + d^q).$$

- $PGL_2(\mathbb{F}_{q^2})/PGL_2(\mathbb{F}_q)$ should take care of structural redundancies. Is it enough ?
- Still, do they really behave like random polynomials of the same degree ? If yes, then constant proportion.
- We did not yet fully investigate a few ideas to get (very partial) proofs, for instance in the case where $P(X) = X$ and $\deg h_i = 1$.
  Already in this "easy" case, need non-trivial machinery.

The rank heuristic

Remember the form of the RHS of the relations:

$$\prod_{(\alpha:\beta)\in\mathbb{P}^1(\mathbb{F}_q)} P(X) - m^{-1}\cdot(\alpha:\beta),$$

where m goes through representatives of $PGL_2(\mathbb{F}_{q^2})/PGL_2(\mathbb{F}_q)$.

Fact

All the systems to solve during the algorithm are obtained by taking $\Theta(q^3)$ rows from a matrix $\mathcal{H}$ of size $(q^3+q)\times(q^2+1)$, that depends only of q.

We label each column by an element of $\mathbb{P}^1(\mathbb{F}_{q^2})$.
Each row corresponds to a matrix m, where we put 1's to describe the image of $\mathbb{P}^1(\mathbb{F}_q)$ by m^{-1}.

The rank heuristic (cont'd)

Theorem

For any ℓ coprime to $q^3 - q$, the matrix $\mathcal{H}$ has full-rank modulo ℓ.

Proof: Can be done with elementary arguments (write appropriate combinations of rows).

Alternate proof:
$\mathcal{H}$ is the incidence matrix of a 3-$(q^2 + 1, q + 1, 1)$ combinatorial **design** called **inversive plane**.
Results in the literature give the eigenvalues of $\mathcal{H}$ over $\mathbb{Q}$.

Question: Is there anything in the design theory literature that could lead to results like: *Any constant proportion of the rows of $\mathcal{H}$ yield a full-rank matrix* ?

Conclusion

Looking back:

- 30 years ago, first $L(1/3)$ DL algorithm by Coppersmith;
- It took more than a decade to get this complexity for a wide range of scenarios;
- Still recent progress on $L(1/3)$-algorithms.

Interesting times!

- We are entering a better-than-$L(1/3)$ era;
- A lot of theoretical and practical improvements are expected in the next few months / years;
- At the moment, absolutely no clue how to extend the quasi-polynomial complexity to large characteristic, or to remove the "quasi".

slide 42

An additional slide to get 42.