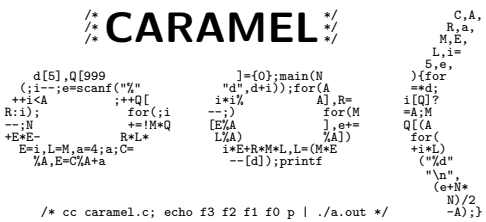
Algorithmes de calcul de logarithme discret dans les corps finis

Grenoble, École de printemps C2 2014

E. Thomé



Mar. 20th-21st, 2014

Part 1

Lecture 1/3: Context and old algorithms

Context, motivations

Exponential algorithms

The archetypal sub-exponential algorithm

Folklore analysis tools

Better algorithms

The «extremely simple» FFS setting

Plan

Finite field DLP

In a cyclic group:

- $(g, x) \to g^x$ is easy: polynomial complexity;
- $(g, g^x) \to x$ is (often) hard: discrete logarithm problem.

Cryptographic applications rely on the hardness of the discrete logarithm problem (DLP).

Application context 1 for FF – DLP

Alice and Bob may settle on $\mathbb{F}_q^*$ as a group to do crypto with.

- Diffie-Hellman (DH), as originally stated.
- Many DLP-based protocols (DSA, electronic voting, ...).

The are smarter choices: elliptic curves (ECDH, ECDSA), ...
Yet finite field DLP does find uses.

Another view on the DLP

In case the group we are working on is not itself cyclic, DLP is defined in a cyclic sub-group (say of order n).

Thus we know that we have an isomorphism.

$$\begin{array}{rcl} \mathbb{Z}/n\mathbb{Z} & \to & \mathsf{G}, \\ k & \mapsto & g^k \end{array}$$

DL is the direction $\leftarrow$. This map is not computationally explicit.

DH versus DL

There are several ancillary problems to the DLP, the closest ones being the (computational) Diffie-Hellman problem:

$$\text{CDH:} \quad (g, g^x, g^y) \to g^{xy}$$

and the decisional Diffie-Hellman problem:

$$\text{DDH:} \quad (g, g^x, g^y, h) \to h \stackrel{?}{=} g^{xy}$$

- Trivial: can solve DLP $\Rightarrow$ can solve CDH $\Rightarrow$ can solve DDH.
- The converse is true for a fairly large class of groups [Mau94, MW96, MSV04].

What we won't do

We do not address here the DH-related problems, but really the computation of DL.

Other ancillary problems

Cryptographic protocols rely on the hardness of some specific problems.

- DH key exchange relies on CDH;
- El Gamal encryption relies on CDH;
- ...

Many newly invented cryptographic protocols come with their new purportedly hard problem.

- In all (DL-related) cases, solving DL is enough to break.
- In many cases, this is the best solution known... not always.

Hardness

A family of groups $\mathcal{G}$

Size parameter $\lambda \longrightarrow \mathcal{G}(\lambda) =$ a set of groups of size λ.

Here, size = size of elements (number of bits).
Some basic requirements:

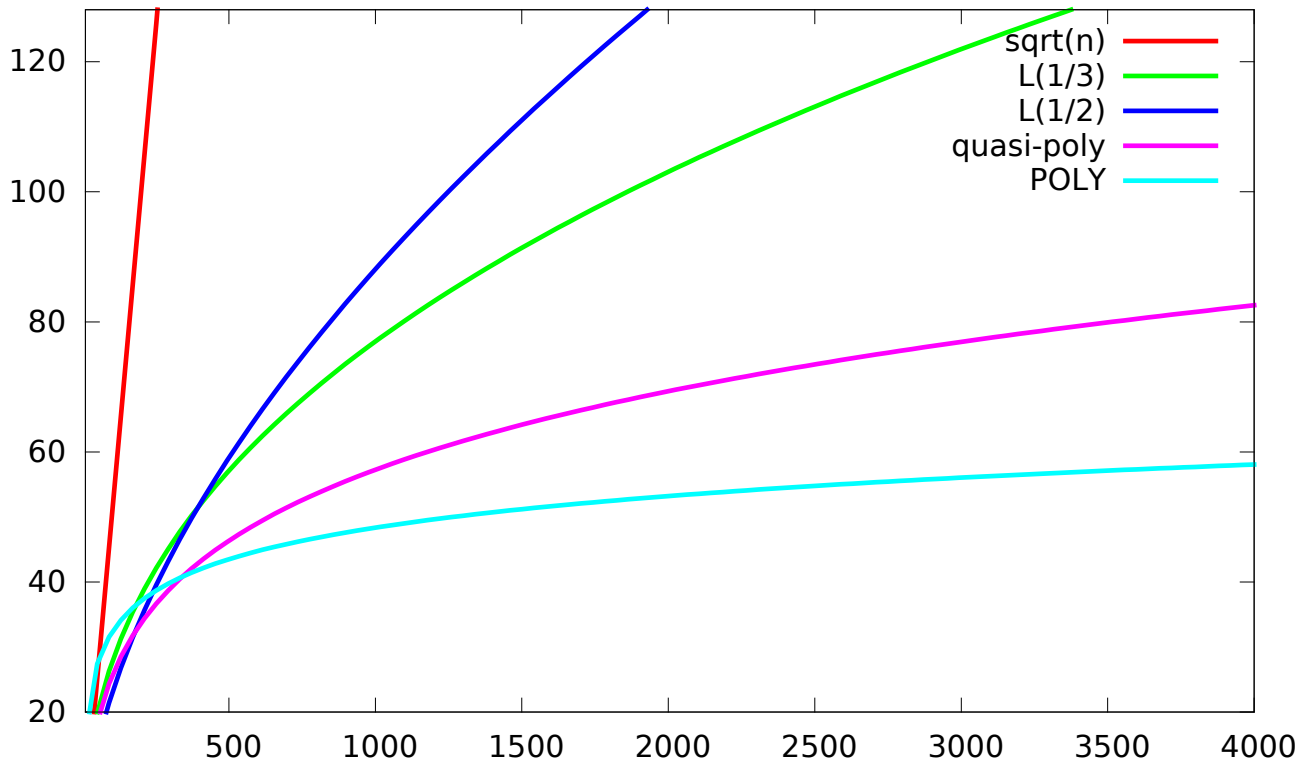
- Counting $n = \#G$ for $G \in \mathcal{G}(\lambda)$ must be feasible;
- We want $\lambda = O(n)$;
- Arithmetic in G must be reasonably efficient.

Examples: finite fields; binary finite fields; prime fields; elliptic curves over binary fields; ...

Hardness: we want the cost of computing DL by most efficient means known to grow fast.

Hardness w.r.t input size

Plotting log(computational difficulty) as a function of $\log(\#G)$.

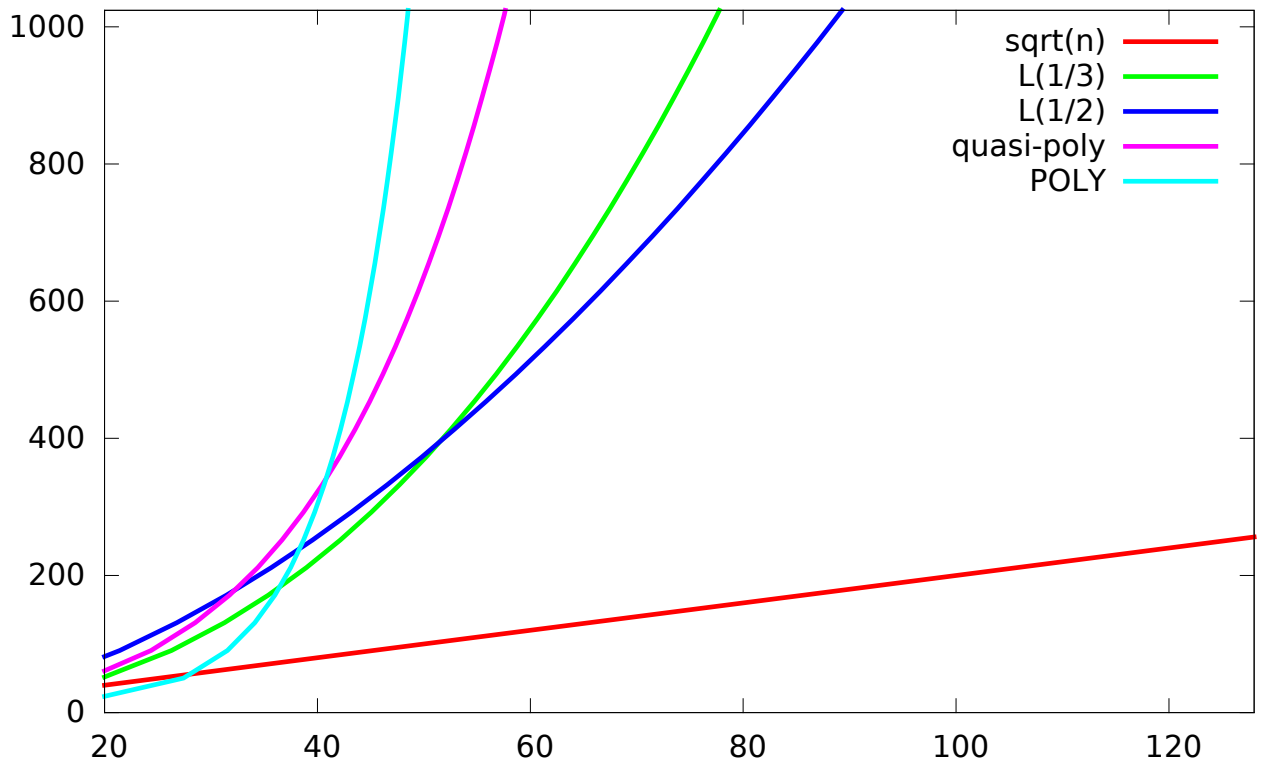


Slow algorithms get expensive pretty soon.

Key size w.r.t desired security / time

Feasibility limit = computational resources; grows with time (Moore's law).

- security parameter=log(attack cost) is a measure of time.
- key size to be chosen appropriately.



The Pohlig-Hellman reduction

Why do we care about counting $n = \#G$?

Assume we have $n = \prod_i p_i^{\alpha_i}$.

PH works on the $\mathbb{Z}/n\mathbb{Z}$ side of the isomorphism seen earlier.

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

- Let $p^\alpha \,||\, n$ and $m = n/p^\alpha$. Then $b = a^m$ is in the cyclic group of order p^α generated by g^m. We can find the log of b in this group, which yields x mod p^α.
- To compute the log in a group of order p^α, one just has to compute a log in a group of order p, and do so α times.

Cost: $\mathrm{DL}(n) = O(\max(\mathrm{DL}(p)))$.

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

Which groups shall we avoid ?

Some groups are bad for cryptography: $(\mathbb{Z}/N\mathbb{Z}, +)$. DL easy !

Some groups have a not-so-hard DL: finite fields, curves of very large genus, class groups of number fields.

Some groups have hard DL: elliptic curves, curves of genus 2.

And of course, beyond that, groups whose order is a product of small primes are a waste !

Reach of studying FF DLP

Finite fields are not ideal for crypto. Why is it interesting at all ?

- It's a basic object. Is worth looking at mathematically speaking.
- It's a basic object. It's been there from the start.
- It's a basic object. Some crypto protocols prefer these.
- Pairings.
- Tricks with subgroups of the multiplicative group.

Pairing-based crypto

Application context 2 for FF – DLP

Some ($\pm$ exotic) cryptographic protocols use pairings:

$$e : \mathbb{G} \times \mathbb{G} \to K^*, \quad (K \text{ finite field})$$

- Identity-based encryption [BF01],
- Tripartite DH [Jou00],
- Short signatures [BLS04],
- ...

The finite field is intrinsically there; cannot be replaced.

Known practical situations:
- $\mathbb{G}$ =an elliptic curve over $\mathbb{F}_q$,
- K a small extension of $\mathbb{F}_q$.

Why do subgroup DLP ?

There's a small catch with the PH reduction.

- Computing the log in G (thus modulo n) might use some special structural properties of G (e.g. stability under some extra operations).
- Computing the log mod p (hence in a subgroup) might fail to exploit these properties.

Subgroups of finite fields

Consider $G = \mathbb{F}_p^\times$, and H a subgroup, $\#H = q \mid (p-1)$.

- We'll see algorithms for DLP in G. Those mandate that $\#G > 2^{1024}$ (so to say).
- Practical attacks for DLP in H, beyond DLP in G, are generic ones. The condition on $q = \#H$ is less stringent.

Subgroups of finite fields

Some protocol designs exploit this subgroup trick.

- DSA takes a $q \approx 160$-bit subgroup of $\mathbb{F}_p^\times$, with 1024-bit p. Signature data is mod q, not mod p.
- The XTR cryptosystem [LV00] uses a reduced representation of elements within a subgroup of $\mathbb{F}_{p^6}$.
- Torus-based cryptography [RS08]: generalization of similar principle.

What this implies for DLP challenges:

- $\#\mathbb{F}_{2^{809}}^\times = p_{201} \times p_{607}$; DLP in the 607-bit subgroup is hardly relevant for crypto.
- Pairing-based crypto example $E \times E \to \mathbb{F}_{q^k}$.
  Only DLP in subgroup of order $\#E$ is interesting.

Subgroups (summary)

Bear in mind

- we are not necessarily interested in the DLP in the whole multiplicative group per se;
- we shall not be annoyed by some uninteresting prime factors behaving oddly.
- DL modulo small primes can be handled differently anyway.

The name of the game

Context for this course

$K = \mathbb{F}_{p^n}$ a finite field; $G = \langle g \rangle$ a subgroup of $K^\times$.

We'll see:

- Basic algorithms which apply to general groups;
- Algorithms specialized for the finite field context.

We might regard the DLP in two possible ways.

- Given $a = g^x \in G$, compute x; just once.
- Do some precomputation, producing data so as to be able to compute many log at smaller cost.

There's a preference for the latter.

The art of FF – DLP

Algorithms specialized for finite field DLP (mostly lecture 2) share background with:

- Algorithms for factoring integers.
  Much of the work which led to DLP algorithms initially came from the factoring business.
- DLP computation of high genus curves [EGT11].

Plan

Context, motivations

Exponential algorithms

The archetypal sub-exponential algorithm

Folklore analysis tools

Better algorithms

The «extremely simple» FFS setting

Generic groups

First focus on DLP algorithms for generic groups. This means we cannot use specific properties of G, just group operations. The group acts as a black box.

Known generic solutions to DLP (here $n = \#G$):

- Enumeration: $O(n)$;
- Shanks (Baby-step–Giant-step):
  deterministic time and space $O(\sqrt{n})$;
- Pollard ρ:
  probabilistic time $O(\sqrt{n})$, space $O(1)$ elements of G.
- Parallel collision search (λ): same context, in parallel.

These are exponential algorithms in the bit-size of G.

The Baby-step–Giant-step method

One writes the $x = \log_g a$ as:
$x = cu + d, \quad 0 \leq d < u, \quad 0 \leq c < n/u$

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

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

Step 2 **(giant steps)**:
- compute $f = g^{-u} = 1/g^u$;
- for $c = 0..n/u$, if $af^c \in \mathcal{B}$, then stop.

End: $af^c = g^d$ for some d, hence $x = cu + d$.

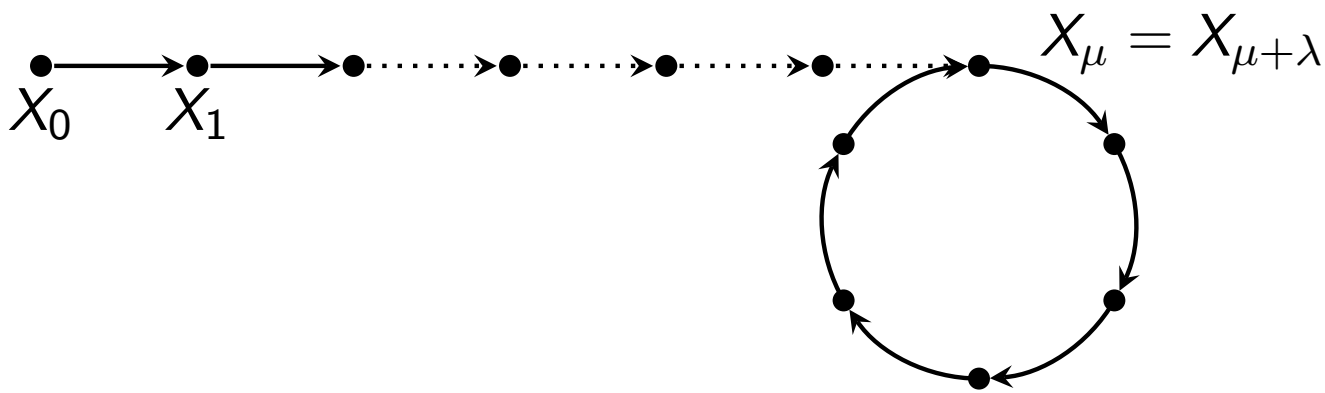
Analysis:
- $C_o = u + n/u$ group operations;
- $C_m = n/u$ membership tests.

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

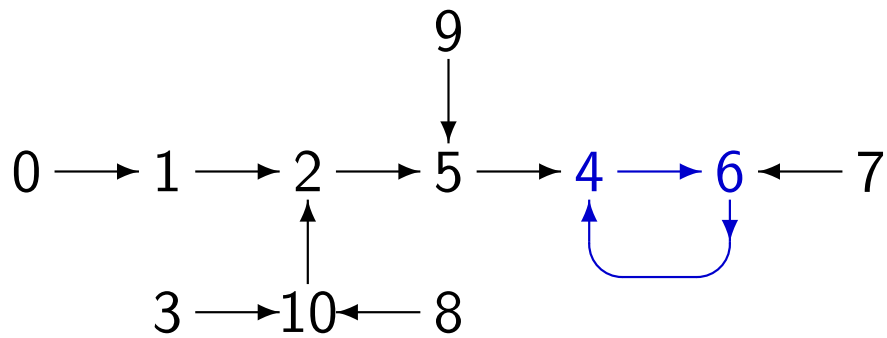
Implementation:
- use hashing to test membership in $\mathcal{B}$;
- all kinds of trade-offs possible if low memory available.

Pollard's ρ

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



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



Application to discrete logarithms

Want to compute $\log_g h$.

A way to create « random-looking » functions

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

Experimentally, for $r \geq 20$, we get f « sufficiently random ».

By iterating $x_{k+1} \leftarrow f(x_k)$, one keeps track of:

- $x_k \in G$;
- Integers u_k and v_k in $\mathbb{Z}/n\mathbb{Z}$ such that $x_k = g^{u_k} h^{v_k}$.

Whenever $x_m = x_n$, we unveil $\log_g h$ (with good probability).

Analyzing Pollard's ρ

Asymptotics of μ and λ [FO90]

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

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

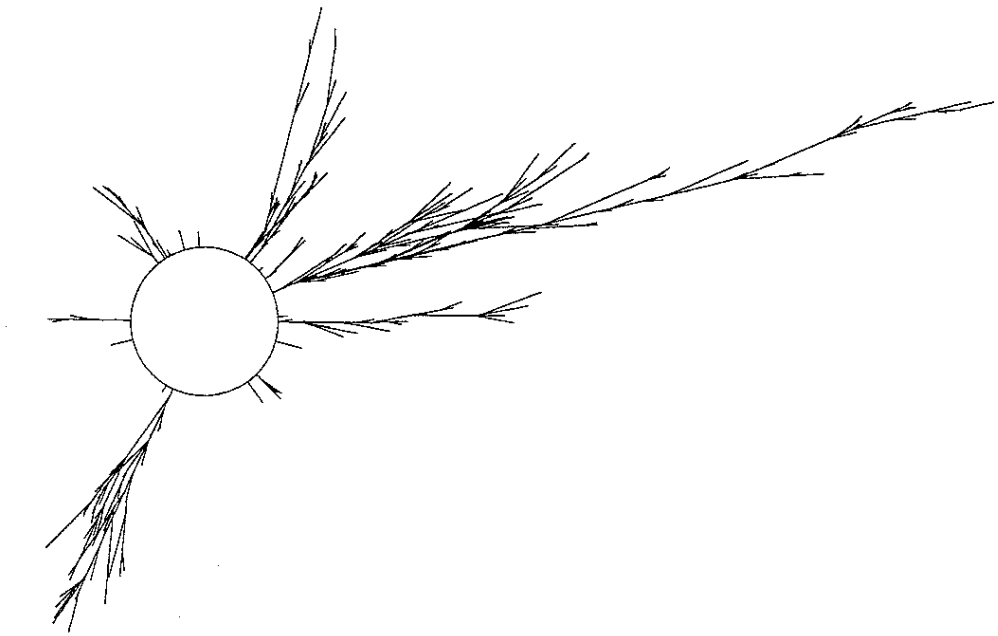
Thm. $\overline{e} \sim \sqrt{\frac{\pi^5 m}{288}} \approx 1.03\sqrt{m}$.

Floyd's algorithm:

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

Searching for collisions in parallel

The « big picture » of the functional graph looks like this:



- Principle: have several workers start at different random points in the graph.
- Challenge: how to detect collisions efficiently ?

The parallel collision search method

Foklore idea, attributed to many ([QD90, vOW99]).

- Node K iterates $x \leftarrow f(x)$.
- Too expensive to check every x will all points computed elsewhere.
- Only rare distinguished points are considered for checking: e.g. if the 20 lowest bits of their hash are zero.
- DPs are stored on a server.
- Caveat: avoid short cycles. Node restart from a new fresh random point quite often.

The complexity in total is still $O(\sqrt{n})$, and this scales up very well.

Is that all ?

We've seen two algorithms which work on generic groups, and solve DL in $O(\sqrt{n})$.

Can we do better ?

Answer: **NO**, not for generic groups.

Nechaev-Shoup [Nec94, Sho97]

For a generic (black-box) group G with $n = \sqrt{G}$, no discrete logarithm algorithm succeeds in time less than $\Omega(\sqrt{n})$.

Caveat: generic groups do not exist.

Computational feats

Generic algorithms are the best ones known for DLP on elliptic curves. Thus the most intensive calculations are in this context.

All used some form of parallel collision search.

- Certicom ECC challenges. Several done by R. Harley ($\sim$2000). Sofware still online.
  `http://cristal.inria.fr/~harley/ecdl_top`
- 112-bit ECDLP using Playstations (2009; LACAL).
  `http://lacal.epfl.ch/112bit_prime`

Computational defeats

Ongoing (?) project: a 131-bit elliptic curve group (with some nice properties).

- Awful lot of computational work;
- All fun stuff for optimizing code has been done, and leaves little place for extra fun;
- Probably not exciting enough.

It seems that the computation is stopped.

`http://ecc-challenge.info/`

Plan

Context, motivations

Exponential algorithms

The archetypal sub-exponential algorithm

Folklore analysis tools

Better algorithms

The «extremely simple» FFS setting

Combination of congruences

As it turns out, finite fields have faster DLP algorithms than generic groups.

- First algorithm of this kind: Adleman [Adl79], based on the combination of congruences idea (cf factoring).
- Analysis requires some standard analytic number theory tools.
- More advanced algorithms followed.

Smoothness

We often talk about smoothness of polynomials or integers.

Definition: smoothness

- A polynomial of degree at most n is k-smooth if all its irreducible factors have degree at most k.
- An integer $\leq X$ is B-smooth if all its prime factors are $\leq B$.

Probability of smoothness is controlled essentially by the size ratio: $u = \frac{n}{k}$, or $u = \frac{\log X}{\log B}$.

Simple-to-remember: the probability is roughly $u^{-u(1+o(1))}$.

- Canfield-Erdős-Pomerance + many variants.
- Some caveats with limit cases.
- Analytic number theory can give much stronger results.

About smoothness

The cost of smoothness testing depends on which kind of object we're talking about.

- Easy for polynomials, just like factoring polynomials, has polynomial complexity.
- For integers, it is trickier, as factoring integers is hard. It's possible to work around this difficulty both theoretically and in practice.

Adleman's algorithm

Consider $G = \mathbb{F}_p^\times$, for an n-bit prime p.
Elements of G can be represented as integers in $[0, p-1]$.
Goal: compute $\log_g h$.

Let $B \approx 2^\beta$ be a bound. Precomputation step.

- Compute random elements $a = g^r$.
  As an integer, a is sometimes B-smooth.
- We thus collect relations $g^r = \prod p_i^{e_i}$ involving a limited set of prime factors p_i.

$$r \equiv e_1(\log p_1) + \cdots + e_K(\log p_K) \mod (p-1).$$

- By linear algebra we solve for the unknowns $(\log p_i)$.

Next, consider the challenge h:

- Compute random elements $b = hg^r$ until b is B-smooth.
- Use the precomputed $(\log p_i)$'s to express $\log h$.

The linear algebra system showing up

Characteristics of the system

- Equations are $r \equiv e_1(\log p_1) + \cdots + e_K(\log p_K) \mod (p-1)$.
- Unknowns are $(\log p_i)$.
- Matrix coefficients are e_i.
- Each matrix row has few coefficients, which are small integers.
- The system is defined over $\mathbb{Z}/(p-1)\mathbb{Z}$.

The algorithmic tools to solve such systems are sparse linear algebra algorithms (which are much more developed now than they were in 1970's).

- Will tell more about these later;
- Good to know: nowadays we do have the tools to do this in quadratic time.

Size matters

Recall $p \approx 2^n$ and $B \approx 2^\beta$.

The set of primes below B is called the factor base.

Which size for the factor base elements ? (= which B ?)

- too small (= too few)?
  - Will have a hard time finding relations.
  - Cheap linear algebra.
- too large (= too many)?
  - Relations become cheaper.
  - Linear algebra becomes an obstacle.

Analysis

Recall $p \approx 2^n$ and $B \approx 2^\beta$.

- The number of primes below B is $B/\log B$, not that far from $B = 2^\beta$.
- $a = g^r$ is an n-bit integer.
  Smoothness probability: $(n/\beta)^{-n/\beta(1+o(1))}$.
  We need 2^β relations.
- Linear algebra cost is polynomial in B.

Optimal choice: $\beta \approx \sqrt{n}$.

The overall complexity is $\exp(O(\sqrt{n \log n}))$.

This is called a sub-exponential complexity.

Consequence: key size = square of time (see slide 10).

Precomputation vs individual logarithms

Note: as presented, Adleman's algorithm has:

- a precomputation stage: logs of FB elememts.
- an individual log stage: given a, find $\log_g a$.

One can show that the latter is cheaper.

Nice DL algorithms nowadays keep this small per-logarithm cost.

Adaptation to binary fields

Obvious fact: what we have just seen for $\mathbb{F}_p$ is also valid for

$$\mathbb{F}_{2^n} = \mathbb{F}_2[x]/P(x)$$

where smoothness of integers is replaced by smoothness of polynomials, and bit size of integers by degree.

Improvements

There have been some improvements of Adleman's algorithm in the early 1980's [BFHMV84].

Useful for later: instead of attempting to factor the integer $a \in [0, p-1]$, use Euclid's algorithm to write

$$a \equiv u/v \mod p$$

with $u \approx v \approx \sqrt{p}$.

- Test smoothness for two half-size numbers instead of one;
- This yields an improvement in practice.

Early 1980's: $\mathbb{F}_{2^{127}}$ was doable with this improved Adleman's algorithm.

Plan

Context, motivations

Exponential algorithms

The archetypal sub-exponential algorithm

Folklore analysis tools

Better algorithms

The «extremely simple» FFS setting

The L function

Handy function introduced (we think) by R. Schroeppel:

$$L_x[\alpha, c] = \exp\left(c(\log x)^{\alpha}(\log\log x)^{1-\alpha}\right).$$

- $L_x[0, c] =$ polynomial in $\log x$.
- $L_x[1, c] =$ exponential in $\log x$.
- L is often called the sub-exponential function.

Usual shorthand in these slides: $L_x[\alpha]$ denotes $L_x[\alpha, c]$ for some explicitly computable constant c.

L function arithmetic

Computation rules with $L_x(\alpha, c)$

$$L_x[a,u] \times L_x[b,v] = \begin{cases} L_x[a, u+o(1)] & \text{if } a > b, \\ L_x[b, v+o(1)] & \text{if } b > a, \\ L_x[a, u+v] & \text{if } a = b. \end{cases}$$

$$L_x[a,u>0] + L_x[b,v>0] = \begin{cases} L_x[a, u+o(1)] & \text{if } a > b, \\ L_x[b, v+o(1)] & \text{if } b > a, \\ O(L_x[a, \max(u,v)]) & \text{if } a = b. \end{cases}$$

$$L_{L_x[b,v]}[a,u] = L_x[ab, uv^a b^{1-a} + o(1)].$$

$$L_x[b,v]^{\log_{\log x} L_x[a,u]} = L_x[a+b, uv].$$

Restating CEP

Reformulation of Canfield-Erdős-Pomerance

An integer $\leq L_x(\alpha, u)$ is $L_x(\beta, v)$-smooth with probability:

$$L_x(\alpha - \beta, -\frac{u}{v}(\alpha - \beta))^{1+o(1)}.$$

For example, a random integer modulo N has a probability $L_N(1/2, \cdot)$ of being $L_N(1/2, \cdot)$-smooth.

Smoothness for integers

Interlude: the ECM factoring method unveils a factor p of N in time $L_p[1/2]$. (Idea: set $B_1 = L_p[1/2, c]$. Hope for $\#(E \bmod p)$, which is $L_p[1, 1]$, to be B_1-smooth.)

Consequence: testing a number $x \approx N$ for smoothness with respect to a bound $B = L_N[\gamma, c]$ costs:

$$L_{L_N[\gamma,c]}[1/2] = L_N[\gamma/2].$$

This may be seen as mostly of theoretical interest, but does save the day when analyzing difficult steps of advanced algorithms.

Analyzing Adleman again

Integers mod p have magnitude $L_p(1,1)$.

Write the smoothness bound B as $L_p[\gamma, c]$ for some γ, c.

- Linear system cost: $L_p[\gamma, 2c]$;
- Smoothness probability: $L_p[1-\gamma, 1/c(1-\gamma)]$.
- Relations needed: $L_p[\gamma, c]$.
- Overall cost: $L_p[\max(\gamma, 1-\gamma)]$.

This readily gives the optimal $\gamma = 1/2$ (and the lower order term comes, too).

All the subexponential algorithms nowadays are analyzed with this machinery.

What hinders Adleman's algorithm

The following points have been improved in further works:

- Elements created as $g^r \bmod p$ are as large as p.
  All we can say is $a \in L_p[1]$.
- For computing individual logarithms (e.g. when we consider the target h), we have no way to take advantage of the fact that h might perhaps have not-so-large bit size.

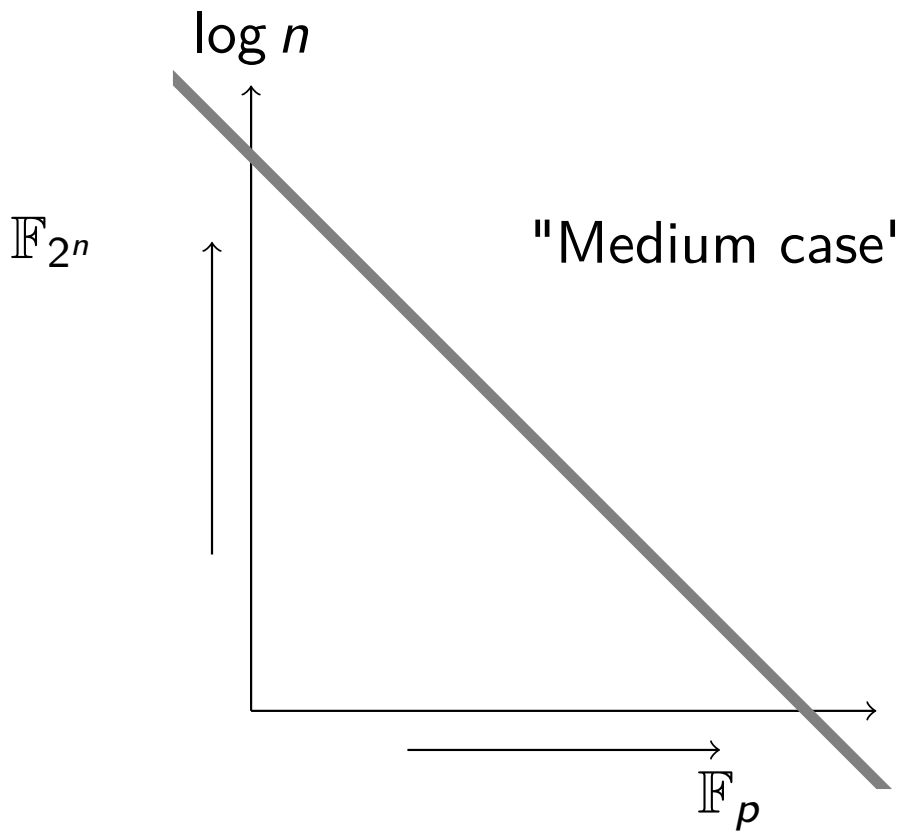
If we could do the latter, we might imagine a recursive procedure:

Large $h \to$ product of smaller elements $\to$ yet smaller $\to \{p_i\}$.

Such a recursive procedure exists in the modern context and is called a descent.

Plan

Finite fields $\mathbb{F}_{p^n}$ w.r.t. p and n



Fields with similar bitsize
as we have $\log(n \log p) \approx$ constant

Algorithms for DLP in $\mathbb{F}_{p^n}$

«Modern» algorithms have improved the complexity somewhat.

The complexity depends much on how n compares to $\log p$.

- DLP in $\mathbb{F}_{p^n}^\times$ for $\log p$ small compared to n: [Cop84], and the Function Field Sieve (FFS) [Adl94, AH99, JL02]:

$$L_q(1/3, 1.53).$$

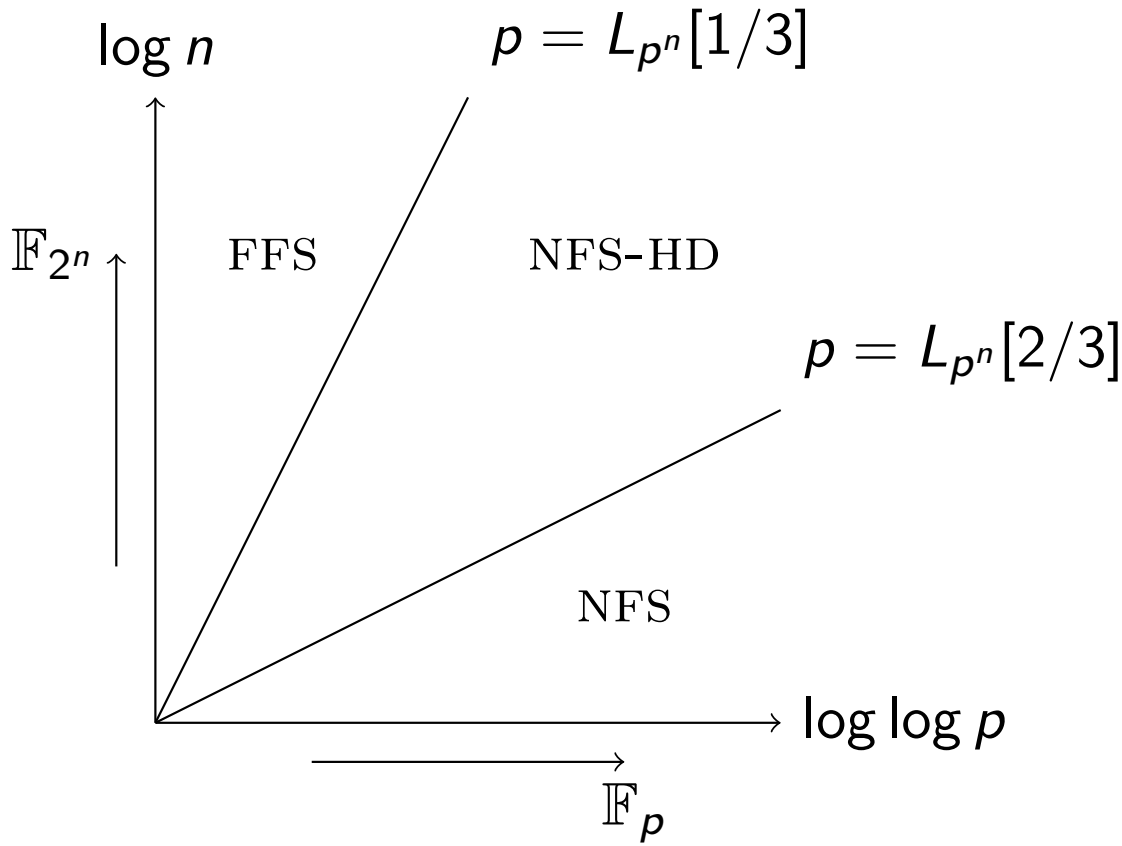
- DLP in $\mathbb{F}_{p^n}^\times$ for n small compared to $\log p$: The Number Field Sieve for DLP (several successive variants [Gor93, Sch99]):

$$L_q(1/3, 1.93).$$

- DLP in $\mathbb{F}_{p^n}^\times$ for fields inbetween: NFS-HD [JLSV06] + some recent improvements in 2014 (Joux, Pierrot):

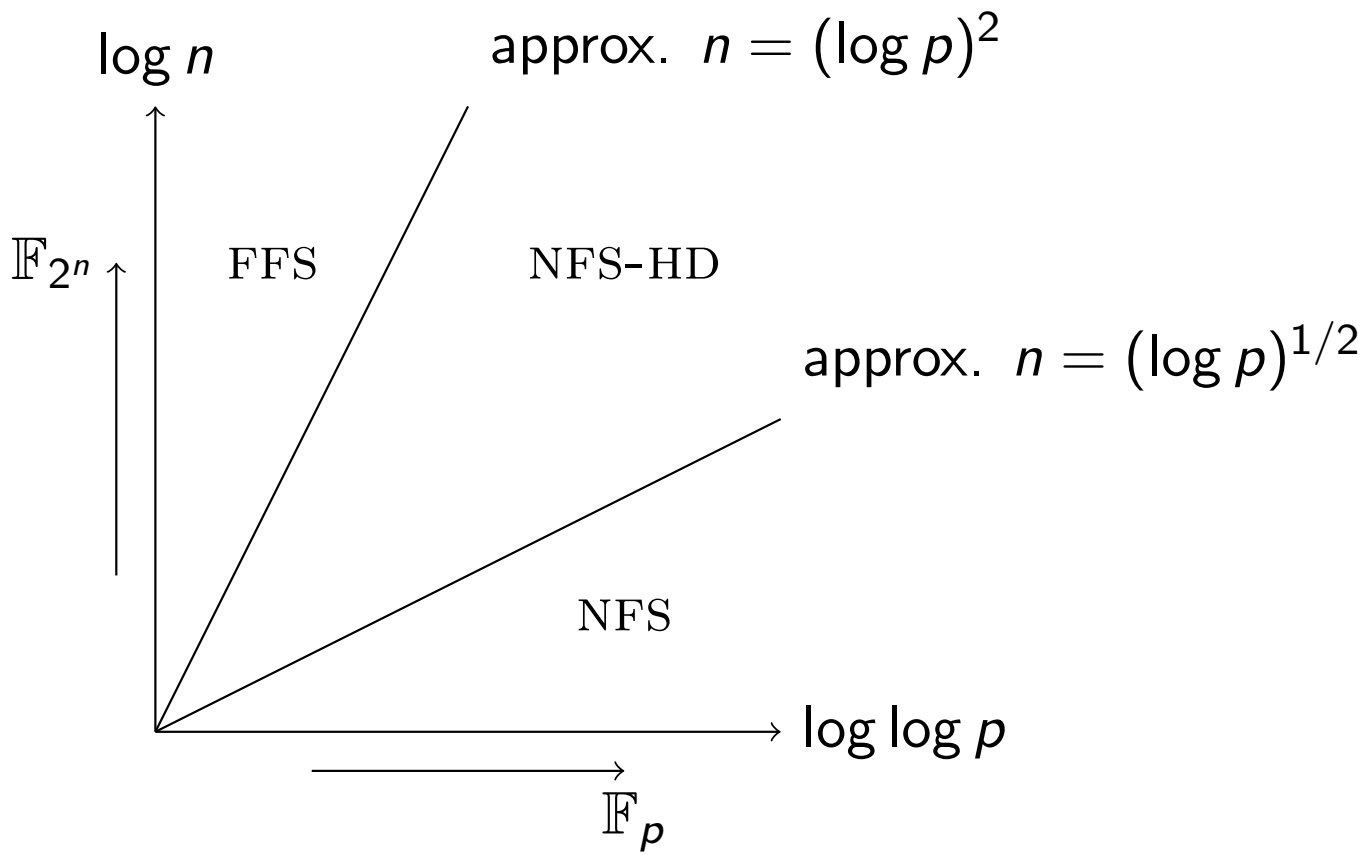
$$L_q(1/3, c) \quad \text{(for some } c\text{)}.$$

Situation in 2006



Lecture 2: the algorithms in this picture;
Lecture 3: how things have changed in the small characteristic case in 2013.

Situation in 2006



Lecture 2: the algorithms in this picture;
Lecture 3: how things have changed in the small characteristic case in 2013.

Plan

Context, motivations

Exponential algorithms

The archetypal sub-exponential algorithm

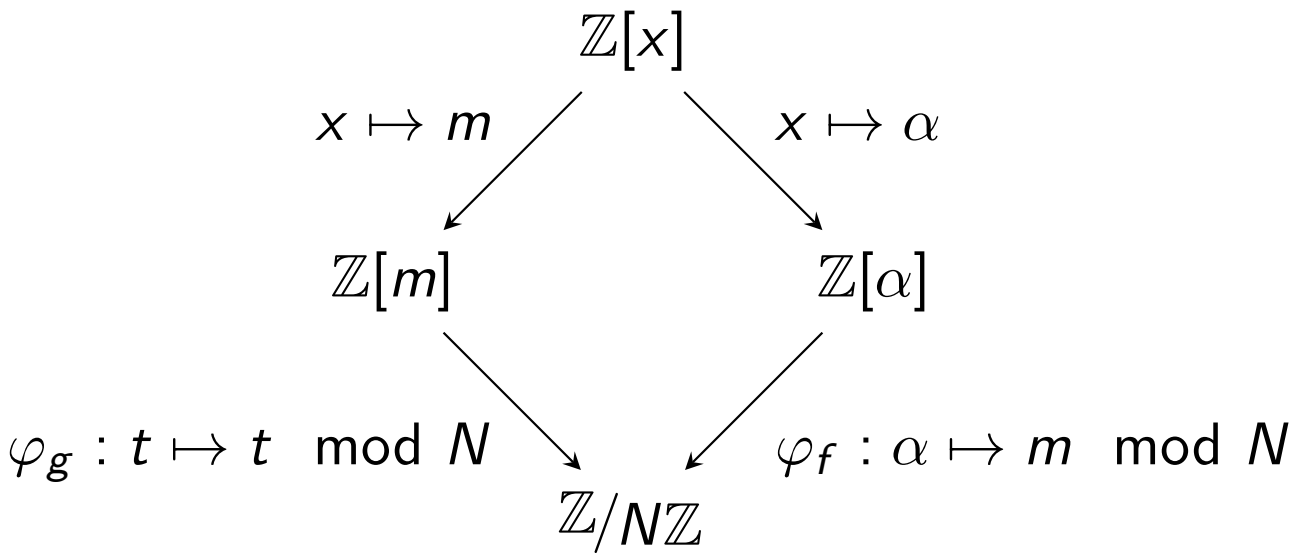
Folklore analysis tools

Better algorithms

The «extremely simple» FFS setting

The NFS-DL setting

This commutative diagram is from the Number Field Sieve as a factoring algorithm.



This is still relevant, but before going into detail, we'll look at a specialized example.

Setting

Let 2 be a small prime (read: works for 3 as well).

Consider two polynomials $f(x)$ and $g(y)$ over $\mathbb{F}_2$ such that

- $\mathrm{Res}_y(y - f(x), x - g(y))$ has an irreducible factor of degree n;
- $\deg f \approx n^{2/3}$;
- $\deg g \approx n^{1/3}$;

Consider random bivariate polynomials $\phi(x, y) = a(x) - yb(x)$ with $\deg_x a, b \approx n^{1/3}$.

We have:
- $\deg \mathrm{Res}_y(y - f(x), \phi) = \deg \phi(x, f(x)) \approx n^{2/3}$;
- $\deg \mathrm{Res}_x(x - g(y), \phi) = \deg \phi(g(y), y) \approx n^{2/3}$.

Maps

Since $\mathrm{Res}(y - f(x), x - g(y))$ has an irreducible factor of degree n, we have two explicit and compatible mappings:

- $\mathbb{F}_2[x] = \mathbb{F}_2[x,y]/(y - f(x)) \to \mathcal{K}_x = \mathbb{F}_2[x]/\mathrm{Res}_y = \mathbb{F}_{2^n}$;
- $\mathbb{F}_2[y] = \mathbb{F}_2[x,y]/(x - f(y)) \to \mathcal{K}_y = \mathbb{F}_2[y]/\mathrm{Res}_x = \mathbb{F}_{2^n}$;

The fields $\mathcal{K}_x$ and $\mathcal{K}_y$ are both isomorphic to $\mathbb{F}_{2^n}$, by an explicitly isomorphic to eachother.

We have thus two «sides»:

- polynomials in x;
- polynomials in y.

Example

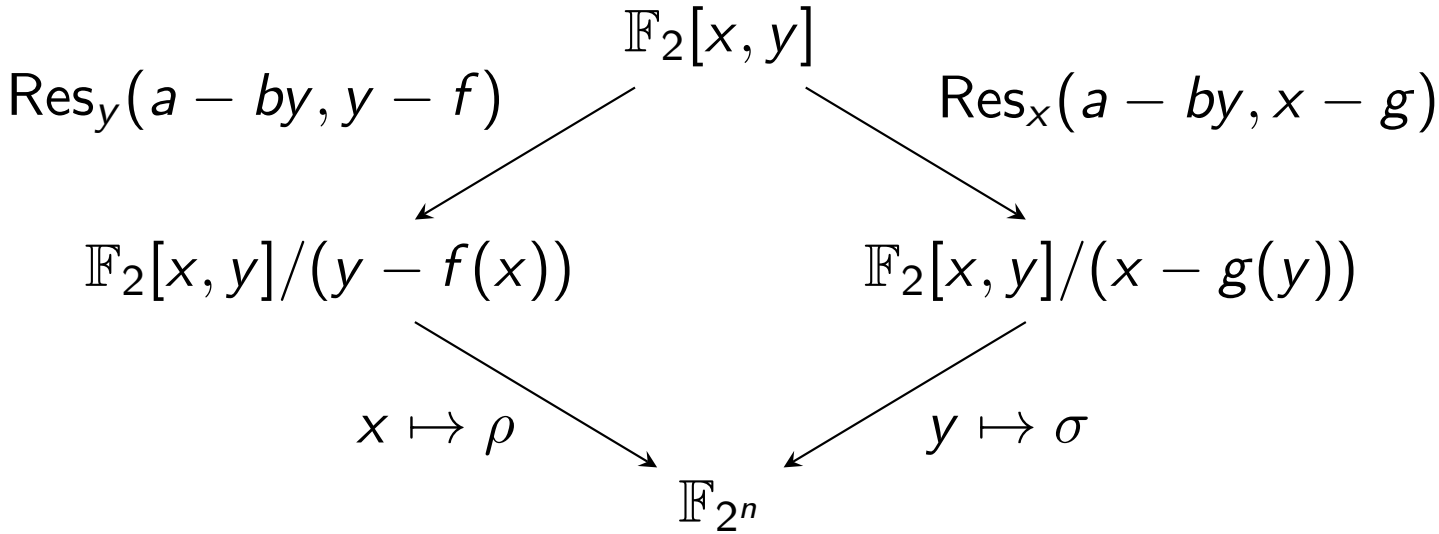
Example for $n = 1000$

We take $f(x) = \boxed{\text{degree 100 in } x}$, $g(y) = \boxed{\text{degree 10 in } y}$.

We have $\phi(x, y) = \underbrace{\boxed{\text{degree 10 in } x}}_{a(x)} - y \times \underbrace{\boxed{\text{degree 10 in } x}}_{b(x)}$;

- $\phi(x, f(x)) = a(x) - f(x)b(x)$ has degree 110 in x;
- $\phi(g(y), y) = a(g(y)) - yb(g(y))$ has degree 101 in y.

The «simple FFS» diagram



where ρ and σ are appropriate representatives in $\mathbb{F}_{2^n}$.

Our two sides are the two sides of the diagram.

We look for favorable $\phi = a(x) - b(x)y$ such that:

- On the x (left) side, $a(x) - f(x)b(x)$ is smooth;
- On the y (right) side, $a(g(y)) - yb(g(y))$ is smooth.

Two sides

We look for favorable $\phi = a(x) - b(x)y$ such that:

- On the x (left) side, $a(x) - f(x)b(x)$ is smooth;
- On the y (right) side, $a(g(y)) - yb(g(y))$ is smooth.

We want BOTH to happen at the same time. Few a, b will work.

Factor bases

Implicitly, we are considering two distinct factor bases:

- Irreducible polynomials in x of small degree ($\leq B$);
- Irreducible polynomials in y of small degree ($\leq B$).

These are distinct because the maps $x \mapsto \rho$ and $y \mapsto \sigma$ have distinct images.

Relations

Consider $\phi = a(x) - b(x)y$ which fulfils the double smoothness condition.

- $a(x) - f(x)b(x) = \prod_i p_i(x)^{e_i}$;
- $a(g(y)) - yb(g(y)) = \prod_j q_j(x)^{d_i}$;
- Because the diagram commutes, we have in $\mathbb{F}_{2^n}$:

$$\prod_i p_i(\rho)^{e_i} = \prod_j q_j(\sigma)^{d_i};$$

$$\sum_i e_i \log(p_i(\rho)) = \sum_j d_j \log(q_j(\sigma)) \mod (2^n - 1);$$

By sparse linear algebra, we can obtain the $\log p_i(\rho)$ and $\log q_j(\sigma)$.

We'll address the descent step in the more general case.

Algorithm

The algorithm follows the usual procedure (similar to Adleman's).

- Initial setup;
- Collect relations between factor base elements;
- Sparse linear algebra to obtain their log;
- Individual logarithms.

Analysis

Collecting all the degree info we have suggested:

- Factor bases = irreducibles of degree $\leq n^{1/3}$; $L[1/3]$
- We have chosen $\deg_x a, b = n^{1/3}$; $L[1/3]$
- The resultants on both sides have degree $n^{2/3}$; $L[2/3]$
- Smoothness probability is thus $L_{2^n}[1/3]$. $L[1/3]$
- Collecting $L_{2^n}[1/3]$ relations takes $L_{2^n}[1/3]$ (just exhausting the search space). $L[1/3]$
- Linear algebra takes $L_{2^n}[1/3]$. $L[1/3]$

The magic here comes from the way we create relations.
This yields a drop from $\frac{1}{2}$ to $\frac{1}{3}$, but no better as is.

References I

[Adl79] L. M. Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, 20th Annual Symposium on Foundations of Computer Science (FOCS '79), 1979, pp. 55–60. San Juan, Puerto Rico, October 29–31, 1979.

[Adl94] _____, *The function field sieve*, ANTS-I, 1994, pp. 108–121. 1st Algorithmic Number Theory Symposium, Cornell University, May 6–9, 1994.

[AH99] L. M. Adleman and M.-D. Huang, *Function field sieve methods for discrete logarithms over finite fields*, Inform. and Comput. **151** (1999), no. 1, 5–16.

[BFHMV84] I. F. Blake, R. Fuji-Hara, R. C. Mullin, and S. A. Vanstone, *Computing logarithms in finite fields of characteristic two*, SIAM J. Alg. Disc. Meth. **5** (1984), no. 2, 276–285.

[BF01] D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Advances in Cryptology – CRYPTO 2001, 2001, pp. 213–229. Proc. 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001.

References II

[BLS04] D. Boneh, B. Lynn, and H. Shacham, *Short signatures from the Weil pairing*, J. Cryptology **17** (2004), no. 4, 297–319.

[Cop84] D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Trans. Inform. Theory **IT–30** (1984), no. 4, 587–594.

[EGT11] Andreas Enge, Pierrick Gaudry, and Emmanuel Thomé, *An $L(1/3)$ discrete logarithm algorithm for low degree curves*, J. Cryptology **24** (2011), no. 1, 24–41.

[FO90] P. Flajolet and A. M. Odlyzko, *Random mapping statistics*, Advances in Cryptology – EUROCRYPT '89, 1990, pp. 329–354. Proc. Eurocrypt '89, Houthalen, April 10–13, 1989.

[Gor93] D. M. Gordon, *Discrete logarithms in $\mathrm{GF}(p)$ using the number field sieve*, SIAM J. Discrete Math. **6** (1993), no. 1, 124–138.

[Jou00] Antoine Joux, *A one round protocol for tripartite Diffie-Hellman*, ANTS-IV, 2000, pp. 385–393. 4th Algorithmic Number Theory Symposium, Leiden, The Netherlands, July 2–7, 2000.

References III

[JL02] Antoine Joux and Reynald Lercier, *The function field sieve is quite special*, ANTS-V, 2002, pp. 431–445. 5th Algorithmic Number Theory Symposium, Sydney, Australia, July 2002.

[JLSV06] Antoine Joux, Reynald Lercier, Nigel P. Smart, and Frederik Vercauteren, *The number field sieve in the medium prime case*, Advances in Cryptology – CRYPTO 2006, 2006, pp. 326–344. Proc. 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006.

[LV00] Arjen K. Lenstra and E. R. Verheul, *The XTR public key system*, Advances in Cryptology – CRYPTO 2000, 2000, pp. 1–19. Proc. 20th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2000.

[Mau94] U. M. Maurer, *Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms*, Advances in Cryptology – CRYPTO '94, 1994, pp. 271–281. Proc. 14th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 21–25, 1994.

References IV

[MW96] U. M. Maurer and S. Wolf, *Diffie-Hellman oracles*, Advances in Cryptology – CRYPTO '96, 1996, pp. 268–282. Proc. 16th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 1996.

[MSV04] A. Muzereau, Nigel P. Smart, and Frederik Vercauteren, *The equivalence between the DHP and DLP for elliptic curves used in practical applications*, LMS J. Comput. Math. **7** (2004), 50–72.

[Nec94] V. I. Nechaev, *Complexity of a determinate algorithm for the discrete logarithm*, Mathematical Notes **55** (1994), no. 2, 165–172.

[vOW99] P. C. van Oorschot and M. J. Wiener, *Parallel collision search with cryptanalytic applications*, J. Cryptology **12** (1999), 1–28.

[QD90] J.-J. Quisquater and J.-P. Delescaille, *How easy is collision search? Application to DES*, Advances in Cryptology – EUROCRYPT '89, 1990, pp. 429–434. Proc. Eurocrypt '89, Houthalen, April 10–13, 1989.

References V

[RS08] Karl Rubin and Alice Silverberg, *Compression in Finite Fields and Torus-Based Cryptography*, SIAM J. Comput. **37** (2008), no. 5, 1401–1428.

[Sch99] O. Schirokauer, *Using number fields to compute logarithms in finite fields*, Math. Comp. **69** (1999), no. 231, 1267–1283.

[Sho97] Victor Shoup, *Lower bounds for discrete logarithms and related problems*, Advances in Cryptology – EUROCRYPT '97, 1997, pp. 256–266. Proc. International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 1997.