Computation of discrete logarithms over finite fields

E. Thomé
Univ. Lorraine, CNRS, Inria, Nancy, France

Jul 20th, 2018

Plan

Introduction

Various positions

Algorithms for key steps and recent computations

Does this really matter ?

Number theoretic problems for crypto

For about 40 years, we've been used to having:

- Integer Factorization (IF)
- and (finite field) discrete logarithms (FF-DLP)

as prominent mathematical problems for public-key cryptography.

Hardness of IF is the security assumption behind RSA, FF-DLP backs Diffie-Hellman, DSA, and others.

Pervasive software assumes these are really hard problems: TLS, SSH, IPsec, ...

Pairing-based crypto

Other application context of FF-DLP

Some cryptographic protocols use pairings.
(ID-based encryption, 3-way DH, Short signatures, ...)

Context:
- E: elliptic curve over $\mathbb{F}_q$.
- $\mathbb{G}_1 = E(\mathbb{F}_q)[r]$, with r prime; $\mathbb{G}_2$: a cousin.
- k: embedding degree.

We have the map:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{F}_{q^k}^*.$$

In this context, study of the DLP in E and in $\mathbb{F}_{q^k}^*$ are equally important. Whichever is weakest jeopardizes security of the protocol.

Pairing-based crypto

Other application context of FF-DLP

Some cryptographic protocols use pairings.
(ID-based encryption, 3-way DH, Short signatures, ...)

Context:
- E: elliptic curve over $\mathbb{F}_q$.
- $\mathbb{G}_1 = E(\mathbb{F}_q)[r]$, with r prime; $\mathbb{G}_2$: a cousin.
- k: embedding degree. $\leftarrow$ this is important!

We have the map:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{F}_{q^k}^*.$$

In this context, study of the DLP in E and in $\mathbb{F}_{q^k}^*$ are equally important. Whichever is weakest jeopardizes security of the protocol.

Motivations beyond crypto

Integer factorization was arguably already a problem with mathematical relevance before RSA was invented.

- First nontrivial algorithmic progress in the 70s (Pollard).

As for FF-DLP, very little seems to predate Diffie-Hellman.

In both cases, the crypto motivation was an excuse to do massive computations.

Algorithms

The same algorithmic setting can be used to factor integers as well as to compute discrete logarithms: the Number Field Sieve.

- Stem of the idea in the late 1980s (Pollard).
- Formulated as a complete algorithm: early 1990s.
- Computing records with NFS started in 1995-1996.
- Many people contributed to the development of NFS.

It is almost one single algorithm with a variety of settings.

Most complexities look the same

Since 2006, DLP in $\mathbb{F}_{p^n}$ costs at most

$$L_Q(1/3, c + o(1)) \text{ for some constant } c$$

where $Q = p^n$ and $L_Q(\alpha, c) = \exp\left(c(\log Q)^\alpha (\log\log Q)^{1-\alpha}\right)$.

Plan

Introduction

Various positions

Algorithms for key steps and recent computations

Does this really matter ?

Context

- We have a finite field. Maybe $\mathbb{F}_p$, maybe $\mathbb{F}_{p^n}$, maybe $\mathbb{F}_{2^n}$.
- We wish to compute discrete logarithms for elements in a multiplicative subgroup of prime order ℓ.
- Typically ℓ is large (160 bits or more).
- To determine the entire DL map for the finite field, proceed piecewise (possibly varying techniques depending on ℓ).

Some handwaving

- We find f with a known root m modulo p.
- Let $\mathbb{Q}(\alpha)$ be the number field defined by f.
- For any polynomial $P(x)$, we have:
  - the integer $P(m)$;
  - the number field element $P(\alpha)$;

  These are compatible: both map to $P(m)$ mod p in $\mathbb{Z}/p\mathbb{Z}$.

$$
\begin{tikzcd}
 & & \mathbb{Z}[x] \arrow[dl, "x \to m"'] \arrow[dr, "x \to \alpha"] & & \\
\boxed{\text{subring of } \mathbb{Q}} & \mathbb{Z}[m] = \mathbb{Z} \arrow[dr, "\text{mod } p"'] & & \mathbb{Z}[\alpha] \arrow[dl, "\alpha \to m"] & \boxed{\text{subring of } \mathbb{Q}(\alpha)} \\
 & & \mathbb{Z}/p\mathbb{Z} & &
\end{tikzcd}
$$

Some handwaving

- We find f with a known root m modulo p.
- Let $\mathbb{Q}(\alpha)$ be the number field defined by f.
- For any polynomial $a - bx$, we have:
  - the integer $a - bm$;
  - the number field element $a - b\alpha$;

  These are compatible: both map to $P(m)$ mod p in $\mathbb{Z}/p\mathbb{Z}$.

$$
\begin{tikzcd}
 & & \mathbb{Z}[x] \arrow[dl, "x \to m"'] \arrow[dr, "x \to \alpha"] & & \\
\text{subring of } \mathbb{Q} & \mathbb{Z}[m] = \mathbb{Z} \arrow[dr, "\text{mod } p"'] & & \mathbb{Z}[\alpha] \arrow[dl, "\alpha \to m"] & \text{subring of } \mathbb{Q}(\alpha) \\
 & & \mathbb{Z}/p\mathbb{Z} & &
\end{tikzcd}
$$

Some handwaving

- We find f with a known root m modulo p.
- Let $\mathbb{Q}(\alpha)$ be the number field defined by f.
- For any polynomial $\prod_i (a_i - b_i x)$, we have:
  - the integer $\prod_i (a_i - b_i m)$;
  - the number field element $\prod_i (a_i - b_i \alpha)$;

  These are compatible: both map to $P(m) \bmod p$ in $\mathbb{Z}/p\mathbb{Z}$.

$$\begin{tikzcd}
 & & \mathbb{Z}[x] \arrow[dl, "x \to m"'] \arrow[dr, "x \to \alpha"] & & \\
\boxed{\text{subring of } \mathbb{Q}} & \mathbb{Z}[m] = \mathbb{Z} \arrow[dr, "\bmod p"'] & & \mathbb{Z}[\alpha] \arrow[dl, "\alpha \to m"] & \boxed{\text{subring of } \mathbb{Q}(\alpha)} \\
 & & \mathbb{Z}/p\mathbb{Z} & &
\end{tikzcd}$$

Write something multiplicative

$$
\begin{tikzcd}
 & \text{group generated by } \mathbb{Z}[x] - \langle p, x-m \rangle \arrow[dl, "x \to m"'] \arrow[dr, "x \to \alpha"] & \\
\text{subgroup of } \mathbb{Q}^* \arrow[dr, "\text{mod } p"'] & & \text{subgroup of } \mathbb{Q}(\alpha)^* \arrow[dl, "\alpha \to m"] \\
 & \mathbb{F}_p^* &
\end{tikzcd}
$$

Write something multiplicative

$$
\begin{tikzcd}
 & \boxed{\begin{array}{c}\text{group generated by}\\ \mathbb{Z}[x]-\langle p, x-m\rangle\end{array}} \arrow[dl, "x \to m"'] \arrow[dr, "x \to \alpha"] & \\
\boxed{\begin{array}{c}\text{subgroup of } \mathbb{Q}^*\\ \text{mod } \ell\text{-th powers}\end{array}} \arrow[dr, "\bmod p"] \arrow[ddr, bend right] & & \boxed{\begin{array}{c}\text{subgroup of } \mathbb{Q}(\alpha)^*\\ \text{mod } \ell\text{-th powers}\end{array}} \arrow[dl, "\alpha \to m"'] \arrow[ddl, bend left] \\
 & \mathbb{F}_p^*/(\mathbb{F}_p^*)^\ell \arrow[d, "\log"] & \\
 & \mathbb{Z}/\ell\mathbb{Z} &
\end{tikzcd}
$$

Write something multiplicative

$$
\begin{tikzcd}
 & \begin{array}{c}\text{group generated by}\\ \mathbb{Z}[x]-\langle p, x-m\rangle\end{array} \arrow[dl, "x \to m"'] \arrow[dr, "x \to \alpha"] & \\
\begin{array}{c}\text{subgroup of } \mathbb{Q}^*\\ \text{mod } \ell\text{-th powers}\end{array} \arrow[dr, "\text{mod } p"'] \arrow[ddr, bend right] & & \begin{array}{c}\text{subgroup of } \mathbb{Q}(\alpha)^*\\ \text{mod } \ell\text{-th powers}\end{array} \arrow[dl, "\alpha \to m"] \arrow[ddl, bend left] \\
 & \mathbb{F}_p^*/(\mathbb{F}_p^*)^\ell \arrow[d, "\log"] & \\
 & \mathbb{Z}/\ell\mathbb{Z} &
\end{tikzcd}
$$

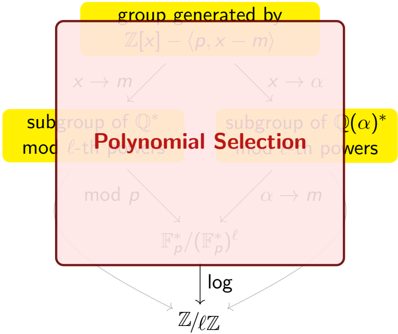
- Two $\mathbb{Z}/\ell\mathbb{Z}$-vector spaces.
- Two compatible linear forms.

We focus on elements on top that give smooth elements on both sides: we then have finite-dimensional vector spaces.

Our two linear forms can be narrowed down by collecting many relations, and solving a large linear system.

Still some work to do afterwards.

Write something multiplicative



- Two $\mathbb{Z}/\ell\mathbb{Z}$-vector spaces.
- Two compatible linear forms.

We focus on elements on top that give smooth elements on both sides: we then have finite-dimensional vector spaces.

Relation collection

Our two linear forms can be narrowed down by collecting many relations, and solving a large linear system.

Linear Algebra

Still some work to do afterwards.

Individual Logarithms (descent)

Finite dimension: how ?

- Elements of $\mathbb{Q}(\alpha)^*$ can be uniquely identified by valuations at prime ideals, and contributions of units.
- Modulo ℓ-th powers: coordinates in $\mathbb{Z}/\ell\mathbb{Z}$.
- In most NFS-like cases, computation of units is intractable. However some arbitrary ℓ-adic character maps can be used, and are equally useful (Schirokauer maps).
- Smoothness condition: only the valuation at a finite number of ideals matters.
  This typically means many ideals, though.
- We identify our linear form by its values at each coordinate (including at Schirokauer maps), usually called virtual logarithms.

Other settings

The NFS framework is quite versatile, and we have several useful variations.

- Straightforward: use two number fields. Pick two number fields $\mathbb{Q}(\alpha)$ and $\mathbb{Q}(\beta)$ with one prime ideal "in common": $\langle p, \alpha - m\rangle$ and $\langle p, \beta - m\rangle$ both prime.
- More interesting: change the base ring $\mathbb{Z}$.

Obstructions are dealt with via Schirokauer maps. Basically the main challenge is to find a setup with appropriate degrees and coefficient sizes.

Note: Most DL-related NFS-like construction are DL-specific because roots modulo p are needed.

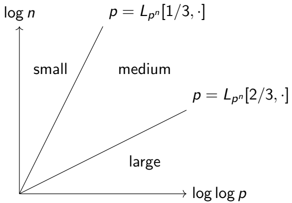
Size of $\log p$ versus n

Different zones

Using $L_{p^n}(\alpha, c) = \exp\left(c(\log p^n)^\alpha(\log\log p^n)^{1-\alpha}\right)$, we define:

- small char.: $p = L_{p^n}(\alpha, c)$ for $\alpha \leq 1/3$ and some c;
- medium char.: $p = L_{p^n}(\alpha, c)$ for $1/3 \leq \alpha \leq 2/3$ and some c;
- large char.: $p = L_{p^n}(\alpha, c)$ for $2/3 \leq \alpha$ and some c.

Some setups are known to work only in select cases, and boundaries are messy.



Complexity in all areas is

$$L_{p^n}(1/3, \gamma + o(1))$$

with various constants γ.

($o(1)$ from Canfield-Erdős-Pomerance.)

A brief timeline

- 1984: Coppersmith, DLP for $\mathbb{F}_{2^n}$. Very first $L(1/3)$ algorithm. In retrospect, does fit in the NFS framework.
- 1990 to 1993: NFS for factoring, for DL mod p.
- 2000 to 2003: better NFS-DL versions.
- 2006: $L(1/3)$ for all finite fields.
- 2013: $\mathbb{F}_{2^n}$ becomes quasi-polynomial.
- 2015-now: More polynomial selection methods for various settings.

An obsolete cousin: the Function Field Sieve

FFS (1994–2013) inherits from NFS. Works in small characteristic.

Consider two plane curves defined over $\mathbb{F}_p$ (p small):

$$\mathcal{C}_1 : \; C_1(x,y) = 0; \qquad\qquad \mathcal{C}_2 : \; C_2(x,y) = 0$$

such that $\mathcal{C}_1 \cap \mathcal{C}_2$ has a point (μ, ν) defining $\mathbb{F}_{p^n}$;
i.e. $\mathrm{Res}(C_1, C_2)$ has an irreducible factor of degree n.

$$\begin{tikzcd}
 & \phi \in \mathbb{F}_p[x,y] \arrow[dl] \arrow[dr] & \\
\text{function on } \mathcal{C}_1 \arrow[dr] & & \text{function on } \mathcal{C}_2 \arrow[dl] \\
 & \text{evaluation on } (\mu,\nu) &
\end{tikzcd}$$

(as before, only the multiplicative diagram is of interest to us)

(obsolete) FFS Setting for $\mathbb{F}_p^n$

$$\begin{tikzcd}
& \mathbb{F}_p[x,y] \arrow[dl] \arrow[dr] & \\
\begin{array}{c}\text{function on } \mathcal{C}_1 \\ \boxed{\text{want smoothness here}}\end{array} \arrow[dr] & & \begin{array}{c}\text{function on } \mathcal{C}_2 \\ \boxed{\text{want smoothness here}}\end{array} \arrow[dl] \\
& \begin{array}{c}\text{evaluation on } (\mu,\nu) \\ \boxed{\text{want relations here}}\end{array} &
\end{tikzcd}$$

- We want smoothness on both sides;
  This occurs in the degree 0 divisor class group of both curves;
- We get relations from the fact that the diagram commutes.

How much complication this means depends on the curves chosen (we *can* make the unit thing trivial here).

NFS for non-prime fields

Define two number fields with polynomials f, g that have a common root in $\mathbb{F}_{p^n}$ ($\deg f, \deg g$ may be $\geq n$).
i.e., both $\mathfrak{p}_f = \langle p, \mu(\alpha)\rangle$ and $\mathfrak{p}_g = \langle p, \mu(\beta)\rangle$ are prime ideals of norm p^n, for the same degree-n polynomial μ.

$$\begin{tikzcd}
 & \mathbb{Z}[x] \arrow[dl, "x \to \alpha"'] \arrow[dr, "x \to \beta"] & \\
K_f = \mathbb{Q}(\alpha) \arrow[dr] & & K_g = \mathbb{Q}(\beta) \arrow[dl] \\
 & \mathcal{O}_{K_f}/\mathfrak{p}_f = \mathbb{F}_{p^n} = \mathcal{O}_{K_g}/\mathfrak{p}_g &
\end{tikzcd}$$

Working setups in medium and large characteristic (JLSV, conjugation, generalized JL; all post-2006).

NFS with towers

Base number field $K_h = \mathbb{Q}[x]/h$ of degree n. Pick two polynomials f, g irreducible in K_h, but with a common root in $\mathcal{O}_{K_h}/p$.

$$\begin{tikzcd}
K_f & & K_g \\
& K_h \arrow[ul, dash] \arrow[ur, dash] & \\
& \mathbb{Q} \arrow[u, dash] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \mathcal{O}_{K_h}[x] \arrow[dl] \arrow[dr] & \\
K_f = K_h(\alpha) \arrow[dr] & & K_g = K_h(\beta) \arrow[dl] \\
& \mathbb{F}_{p^n} &
\end{tikzcd}$$

- First suggested by Schirokauer in 1999.
- Revived in 2015. Works well for large characteristic and $n > 1$.
- The first key to assessing smoothness is the Norm map.

Which construction?

Each method has its own asymptotic complexity, subject to the condition that p and n are within some prescribed scenario.

- We have some "best" constructions,
- and some ties or close calls.

In practice

For a given target finite field, we try out various constructions and easily assess the smoothness probabilities, (heuristically) based on the size of the (absolute) norms.
Important extra topics:

- arrange for f and/or g to favour smoothness (= have many prime ideals of small norms).
- if/when Galois properties can be enforced, nice benefit.

NFS for special primes

Sometimes p has a special form, e.g. given by the evaluation of a low-degree polynomial with small coefficients.

It is *often* possible to take advantage of this.

- Once in a blue moon, p has precisely the ideal kind of expression to allow for a choice of number fields where the typical norms are small on one of the two sides;
- Some pairing-based setups give way to such attacks (= allow for unusually efficient NFS-like DLP computation).
- This is also useful for showcasing ideal situations where algorithms perform well.

Examples: GNFS for factoring

GNFS for factoring N cannot play many tricks because polynomial solving modulo N is not an option.

Typical situation:

- $\deg f = d, \ \deg g = 1$. Coefficients of similar size.
- It is sufficient to search for relations that come from $a - bx$.
- Factoring $\Rightarrow$ linear algebra mod 2.
- Complexity $L_N(1/3, (64/9)^{1/3} + o(1))$.

Also, more variants:

- Non-linear: bi-quadratic and more;
- Multiple number fields (better complexity).

Examples: SNFS for factoring

Historically the very first instance. Adapts to the case where N is a of a very special form that allows a very efficient setup.

Typical situation:

- $\deg f = d$, $\deg g = 1$. Coefficients NOT of the same size:
  - Coefficients of f typically small;
  - Coefficients of g might be larger.
- It is sufficient to search for relations that come from $a - bx$.
- As f is small, the number field computations are more accessible, but it is not really useful.
- Factoring $\Rightarrow$ linear algebra mod 2.
- Complexity $L_N(1/3, (32/9)^{1/3} + o(1))$.

Examples: GNFS for DLP over $\mathbb{F}_p$

Two options:

- Either $\deg f = d,\ \deg g = 1$ similar to factoring;
- Or use Joux-Lercier polynomial selection method which gives $\deg f = d,\ \deg g = d - 1$. (DLP-only!).
- Which one wins depends on the size of p.

In both cases:

- DLP $\Rightarrow$ linear algebra mod ℓ. Harder.
  Typically dominates, but dominates less if ℓ is tiny.
- Search for relations with $a - bx$ is ok, but per-logarithm cost is asymptotically lower if we search for higher-degree functions.
- Complexity $L_p(1/3, (64/9)^{1/3} + o(1))$.
  (multiple number fields variant also exists)

Examples: SNFS for DLP over $\mathbb{F}_p$

Applies when p allows for exceptionally good polynomial selection.

- "ideal" set-up similar to SNFS-factoring.
- Complexity $L_p(1/3, (32/9)^{1/3} + o(1))$.

Examples: NFS for DLP over $\mathbb{F}_{p^n}$

Method	$\deg f$	$\deg g$	$\|f\|_\infty$	$\|g\|_\infty$
Joux-Lercier-Smart-Vercauteren (1)	n	n	$\sqrt{p}$	$\sqrt{p}$
Joux-Lercier-Smart-Vercauteren (2)	$D \geq n$	n	$p^{n/(D+1)}$	$p^{n/(D+1)}$
Generalized Joux-Lercier	$d+1$	$d \geq n$	1	$p^{n/(d+1)}$
Conjugation	$2n$	n	1	$\sqrt{p}$

GJL for large p: $L_{p^n}(1/3, (64/9)^{1/3} + o(1))$.
Conj for medium p: $L_{p^n}(1/3, (96/9)^{1/3} + o(1))$.

Examples: NFS for DLP over $\mathbb{F}_{p^n}$

Method	$\deg f$	$\deg g$	$\|f\|_\infty$	$\|g\|_\infty$
Joux-Lercier-Smart-Vercauteren (1)	n	n	$\sqrt{p}$	$\sqrt{p}$
Joux-Lercier-Smart-Vercauteren (2)	$D \geq n$	n	$p^{n/(D+1)}$	$p^{n/(D+1)}$
Generalized Joux-Lercier	$d+1$	$d \geq n$	1	$p^{n/(d+1)}$
Conjugation	$2n$	n	1	$\sqrt{p}$

GJL for large p: $L_{p^n}(1/3, (64/9)^{1/3} + o(1))$.
Conj for medium p: $L_{p^n}(1/3, (96/9)^{1/3} + o(1))$.

Sometimes better with *tower variants*:

- Plain TNFS $L_{p^n}(1/3, (64/9)^{1/3} + o(1))$.
- Combinations of TNFS + methods above then *n* is composite. At best, $L_{p^n}(1/3, (48/9)^{1/3} + o(1))$.

No one-size-fits-all

We have a much richer situation than for integer factoring.

The variety of setups is such that it is inappropriate to assess the DLP hardness in $\mathbb{F}_{p^n}$ with one single asymptotic formula.

No one-size-fits-all

We have a much richer situation than for integer factoring.

The variety of setups is such that it is inappropriate to assess the DLP hardness in $\mathbb{F}_{p^n}$ with one single asymptotic formula.

Briefly put, it is a real mess.

Even asymptotically, the complexities at the boundary cases are just horrible.

It is also common to include multi-NFS variants of all these analyses. Mildly better constants.

No one-size-fits-all

While $L_Q(1/3, (64/9)^{1/3} + o(1))$ fits the case $Q = p$ (prime fields), it need not be so for larger degree.

- Sometimes harder, sometimes easier.
- Not only different asymptotics.
  Case-by-case practical efficiency differs.
- The case where p is special deserves extra care.
- The size of the subgroup of $\mathbb{F}_{p^n}^*$ also matters because it impacts the linear algebra step.

The bodacious assumption that FF-DLP in $\mathbb{F}_Q$ (with $Q = p^n$) is harder than factoring N for $N \approx Q$ is just wrong.

Security claims of some pairing-based systems are sometimes based on shaky grounds (special p, composite n, small ℓ, ...).

Plan

Introduction

Various positions

Algorithms for key steps and recent computations

Does this really matter ?

Collecting relations

The Number Field Sieve involves sieving.

To search for a, b such that $a - bx$ gives smooth norms on both sides, we can:

- ~~trial divide~~ factor the norms with ECM, keep the smooth ones;
- sieve for a, b. Candidate divisors $\mathfrak{p}$ run through a factor base. special-q sieving + sieving by vectors are key instruments.
- Multiply everything together, and use remainder trees to check for smoothness.

Appropriate scheduling varies. Examples:

- Sieve on both sides up to some bound, finish with ECM.
- Sieve on one side, remainder tree on the other side, then ECM.

Higher degree sieving

Several of the variations of NFS call for sieving not only for good $a - bx$:

- Sometimes higher degree functions are needed;
- Sometimes a and b live in a number field.

The algorithms for higher-degree sieving are not (yet?) as efficient as for 2-dimensional sieving.

See talk by L. Grémy.

Linear algebra

The virtual logarithms are obtained by solving an $N \times N$ sparse linear system defined over $\mathbb{Z}/\ell\mathbb{Z}$. We let γ denote the average row weight (typically < 200).

The block Wiedemann algorithm is commonly used.

- The blocking parameters allow for some flexibility in the organization of the computation.
- Needs $(1 + o(1))N$ matrix-times-vector products.
  Each costs γN additions and N reductions mod p, plus communication (threads / MPI).
- Fast, parallel computation of linear generators for matrix power series is a key step.

2015: the Logjam attack

We learned that the security community lived peacefully under the assumption that 512-bit DLP was fine:

- Because it was harder than factoring. (yet, 2007 record).
- And because anyway, a large computation just to break one session key is no big deal.

512-bit DLP was still offered as a compatibility solution for key exchange by web sites such as `fbi.gov`.

Logjam results

Cryptanalysis: not a large precomputation (10 core-y)
and then roughly one minute per individual log (10 core-mn).

2016: 768-bit DLP

Largest FF-DLP to date for “honest” primes.

Took about 5300 core-years.

Most interesting is the fact that sieving was done only on one side.
The norm on the other side was factored with remainder trees.

RFC5114

```
Network Working Group                                        M. Lepinski
Request for Comments: 5114                                       S. Kent
Category: Informational                                 BBN Technologies
                                                            January 2008


         Additional Diffie-Hellman Groups for Use with IETF Standards


2.  Additional Diffie-Hellman Groups

   This section contains the specification for eight groups for use in
   IKE, TLS, SSH, etc.  There are three standard prime modulus groups
   and five elliptic curve groups.  All groups were taken from
   publications of the National Institute of Standards and Technology,
   specifically [DSS] and [NIST80056A].  Test data for each group is
   provided in Appendix A.

2.1.  1024-bit MODP Group with 160-bit Prime Order Subgroup

   The hexadecimal value of the prime is:

   p = B10B8F96 A080E01D DE92DE5E AE5D54EC 52C99FBC FB06A3C6
       9A6A9DCA 52D23B61 6073E286 75A23D18 9838EF1E 2EE652C0
       13ECB4AE A9061123 24975C3C D49B83BF ACCBDD7D 90C4BD70
       98488E9C 219A7372 4EFFD6FA E5644738 FAA31A4F F55BCCC0
       A151AF5F 0DC8B4BD 45BF37DF 365C1A65 E68CFDA7 6D4DA708
       DF1FB2BC 2E4A4371

   The hexadecimal value of the generator is:

   g = A4D1CBD5 C3FD3412 6765A442 EFB99905 F8104DD2 58AC507F
       D6406CFF 14266D31 266FEA1E 5C41564B 777E690F 5504F213
       160217B4 B01B886A 5E91547F 9E2749F4 D7FBD7D3 B9A92EE1
       909D0D22 63F80A76 A6A24C08 7A091F53 1DBF0A01 69B6A28A
       D662A4D1 8E73AFA3 2D779D59 18D08BC8 858F4DCE F97C2A24
       855E6EEB 22B3B2E5

   The generator generates a prime-order subgroup of size:

   q = F518AA87 81A8DF27 8ABA4E7D 64B7CB9D 49462353
```

Here is p
Here is $q \mid (p-1)$
Please use for crypto.

Supported by:

- 900K (2.3%) HTTPS hosts
- 340K (13%) IPsec hosts

2016: kilobit SNFS

What if the standards designer played the Texas sharpshooter ?

Two possible scenarios. Not sure we can tell them apart.

- p chosen really at random. Attacker wants good NFS setup to attack p.
- Standards is rigged. Agency first chose f and g, and then published p, while f and g are kept secret.

Is there a real possibility to have such a “trapdoor” ?

- Would it be a game changer to the DL computation ?
- Would it be conspicuous ?

Back to 1992

This question was raised long back in 1992.

> So far, it has not been demonstrated that trapdoor moduli for the discrete logarithm problem can be constructed such that a) they are hard to detect, and b) knowledge of the trapdoor provides a quantifiable computational advantage for parameter sizes that could actually be computed by known methods, even with foreseeable machines.
>
> —K. S. McCurley, EC92 panel.

Part of the 1992 discussions focused on why a lower bound on p should be 1024 bits, not 512.

But the above points seemed to suffice to settle the discussion on the trapdoor: too conspicuous, and not a game-changer anyway.

However:

- NFS technology has gone a long way since 1992.
- And we're not speaking of the same parameter range.

Exploiting the trapdoor in the modern era

We generated a target 1024-bit prime in 12 core-hours.

The public part:

$$
\begin{aligned}
p &= 16332398724044367910140207009304915503098943980691751 \\
&\quad 91735800707915692277289328503584988628543993514237336 \\
&\quad 97660534800194492724828721314980248259450358792069235 \\
&\quad 99182658894420044068709413666950634909369176890244055 \\
&\quad 53414932372965552542473794227022215159298376298136008 \\
&\quad 1208200612403808946361023923615765125218049 1 \\
q &= 1120320311183071261988433674300182306029096710473\ ,
\end{aligned}
$$

and the hidden polynomials:

$$
\begin{aligned}
f &= 1155\,x^6 + 1090\,x^5 + 440\,x^4 + 531\,x^3 - 348\,x^2 - 223\,x - 1385 \\
g &= 567162312818120432489991568785626986771201829237408\,x \\
&\quad -663612177378148694314176730818181556491705934826717\ .
\end{aligned}
$$

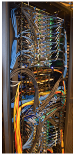
NFS-DL with Cado-NFS

We used Cado-NFS to do the DL computations.

- Complete, LGPL-licensed NFS and NFS-DL implementation;
- developed in Nancy since 2007;
- 14,000 commits. 230,000 lines of C and C++ code;
- Used for several DL records.

Computation timings

Linear algebra was done on higher-end hardware with fast interconnect (Infiniband FDR 56Gbps, Cisco UCS 40Gbps)



Used parameters $m = 24$, $n = 12$ for block Wiedemann.

	sieving	linear algebra			individual log
		sequence	generator	solution	
cores	≈3000	2056	576	2056	500–352
CPU time (core)	240 years	123 years	13 years	9 years	10 days
calendar time	1 month	1 month			80 minutes

Lessons from kilobit SNFS

1024-bit DLP can be easy for an attacker that maliciously chose the prime to his liking.

We found no easy way to prove that a trapdoor is present.

Verifiable randomness is necessary.

- It's not even the question of accusing anyone of wrongdoing. We found no smoking gun.
- But the lack of verifiable randomness is a major hindrance for trust in cryptographic standards.

Of course **people still get it awfully wrong**.

E.g. the standardized French and Chinese elliptic curves are really really bad to this regard.

Plan

Introduction

Various positions

Algorithms for key steps and recent computations

Does this really matter ?

Successors to IF and FF-DLP

Finite fields for crypto look quaint, we have much better!

Elliptic curves ($+$ genus 2):

- part of the the crypto portfolio for years. Hardness of EC-DLP has arguably been a well-studied, mature topic for about 20 years.
- Took off in widely deployed software around 2005–2010. (Your phone most likely does EC-DH).
- BUT...

The quantum threat

Post-quantum crypto:

- Ongoing effort to propose new primitives for standardization;
- No reason to believe that time-to-market is less than 10 to 20 years away.
- Interim suggestion of NSA to not switch to ECs now that they're not the long-term solution they seemed to be.

How to change the world ?

"Compatibility" often works against security:

> "be strict in what you provide,
> be liberal in what you accept"
> ...is not always a good idea

See the different attacks presented !

Hard facts are essential instruments towards getting rid of outdated crypto.

I would rather not like seeing FF-DLP still back a large share of the world's public key crypto in 15+ years when PQ deployment starts being real.

Thanks for your attention