Discrete Logarithm in $\mathrm{GF}(2^{809})$ with FFS

Razvan Barbulescu Cyril Bouvier Jérémie Detrey
Pierrick Gaudry Hamza Jeljeli Emmanuel Thomé
Marion Videau Paul Zimmermann

CARAMEL project-team, LORIA, INRIA / CNRS / Université de Lorraine,
`<first-name>.<last-name>@loria.fr`

PKC 2014, Buenos Aires, March 26th, 2014

Discrete Logarithm Problem

Discrete Logarithm

Given a cyclic group $G = \langle g \rangle$ written multiplicatively, the discrete logarithm of $h \in G$ is the unique k in $[0, \#G - 1]$ s.t.

$$h = g^k.$$

- In certain groups, the discrete logarithm problem (DLP) is computationally hard.
- The inverse problem (discrete exponentiation) is easy.
- Widespread use in **public-key** protocols/implementations:
  - Diffie–Hellman key exchange,
  - ElGamal encryption,
  - DSA signature,
  - pairing-based cryptography, ...

DLP in finite fields of small characteristic

Fields $\mathrm{GF}(p^n)^\times$, with p a small prime (esp. $p = 2$), provide implementation advantages for cryptography.

Before 2013

- **Function Field Sieve** (FFS) algorithm, complexity in $L_{p^n}(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}) = \exp\left(\sqrt[3]{\frac{32}{9}}(\log p^n)^{\frac{1}{3}}(\log\log p^n)^{\frac{2}{3}}\right)$ [Adleman 1994]

After 2013

- $\boldsymbol{L(\frac{1}{4} + o(1))}$ algorithm [Joux 2013] + [Göloğlu et al. 2013]
- **Quasi-polynomial-time** (QPA) algorithm [Barbulescu, Gaudry, Joux, Thomé 2013].

Records:

- $\mathrm{GF}(2^{kp})$: $\mathrm{GF}(2^{6168}) = \mathrm{GF}((2^{24})^{257})$ [05/2013], $\mathrm{GF}(2^{9234}) = \mathrm{GF}((2^{162})^{57})$ [01/2014] using $L(1/4)$ algorithm
- $\mathrm{GF}(2^p)$: $\mathrm{GF}(2^{613})$ [09/2005], $\mathrm{GF}(2^{809})$ [04/2013] using FFS.

Motivations

- Better extrapolation of FFS computational limits:
  - evolution of resources (last record is 8 years old),
  - use of new facilities (GPUs),
  - prepare the ground for FFS in $\mathrm{GF}(2^{1039})$.
- Investigate accelerating critical parts of the FFS algorithm.
- Determine the cut-off points where FFS is surpassed by the new methods (prime-degree extensions?).
- The new algorithms still rely on bits taken from FFS.

Table of Contents

Table of Contents

Index-calculus algorithms

$G = \langle g \rangle$, g of prime order $\ell = \#G$.

Main Idea:

- Collect *relations* of the form $\prod_i \alpha_i^{e_i} = 1$, where the α_i's belong to a predefined subset of G (*factor base*).
- Each relation yields a linear equation in $\mathbb{Z}/\ell\mathbb{Z}$: $\sum_i e_i \log_g(\alpha_i) \equiv 0 \pmod{\ell}$, where the $\log_g(\alpha_i)$'s are the unknowns.

$\rightarrow$ find enough ($\geq$ #factor base) relations.

- Compute the $\log_g(\alpha_i)$'s by solving the corresponding system modulo ℓ.
- Compute $\log_g(h)$, for a given $h \in G$:
  - write $h = \displaystyle\prod_i \alpha_i^{f_i}$.

  $\rightarrow$ $\log_g(h) \equiv \displaystyle\sum_i f_i \log_g(\alpha_i) \pmod{\ell}$.

Function Field Sieve

How to construct $\mathrm{GF}(p^n)$?

- $f, g \in \mathrm{GF}(p)[t][x]$, s.t. $\mathrm{Res}_x(f,g)$ contains an irreducible factor $\varphi(t)$ of degree n.
- $\mathrm{GF}(p^n)$ is therefore obtained as $\mathrm{GF}(p)[t]/\varphi(t)$.

How to find relations?

$$\begin{tikzcd}
 & \mathrm{GF}(p)[t][x] \arrow[dl, "x\mapsto\alpha_f"'] \arrow[dr, "x\mapsto\alpha_g"] & \\
\mathrm{GF}(p)[t][x]/f(x) \arrow[dr, "\alpha_f\mapsto m \bmod \varphi"'] & & \mathrm{GF}(p)[t][x]/g(x) \arrow[dl, "\alpha_g\mapsto m \bmod \varphi"] \\
 & \mathrm{GF}(p)[t]/\varphi(t) &
\end{tikzcd}$$

m the common root modulo φ

Function Field Sieve

How to construct $\mathrm{GF}(p^n)$?

- $f, g \in \mathrm{GF}(p)[t][x]$, s.t. $\mathrm{Res}_x(f, g)$ contains an irreducible factor $\varphi(t)$ of degree n.
- $\mathrm{GF}(p^n)$ is therefore obtained as $\mathrm{GF}(p)[t]/\varphi(t)$.

How to find relations?

$$\begin{tikzcd}
& a(t)-b(t)x \in \mathrm{GF}(p)[t][x] \arrow[dl, "x \mapsto \alpha_f"'] \arrow[dr, "x \mapsto \alpha_g"] & \\
\underset{\text{smooth?}}{a(t)-b(t)\alpha_f} \in \mathrm{GF}(p)[t][x]/f(x) \arrow[dr, "\alpha_f \mapsto m \bmod \varphi"'] & & \mathrm{GF}(p)[t][x]/g(x) \ni \underset{\text{smooth?}}{a(t)-b(t)\alpha_g} \arrow[dl, "\alpha_g \mapsto m \bmod \varphi"] \\
& \mathrm{GF}(p)[t]/\varphi(t) &
\end{tikzcd}$$

- Smooth: an element is B-smooth if its factorization involves only prime ideals whose norms have degree less than or equal to B.
- If doubly smooth, 2 factorizations of $a(t) - b(t)x$ in the 2 “sides” $\rightarrow$ equation between two products of elements of the factor base.

Steps of FFS

1. Polynomial selection: find f and g.
   [Barbulescu and Zimmermann]
2. Relation collection (a.k.a. "sieving"): look for doubly smooth elements
   - **Special-$\mathfrak{q}$ sieving**: sieve on elements whose norm is divisible by a given prime ideal $\mathfrak{q}$ $\implies$ increase the probability that the remaining part is smooth.
   - **Lattice-sieving** for various special-$\mathfrak{q}$'s.

   [Detrey, Gaudry and Videau]
3. Filtering: prepare the linear algebra over $\mathbb{Z}/\ell\mathbb{Z}$.
   [Bouvier and Thomé]
4. Linear algebra: solve a system of linear equations modulo ℓ.
   [J. and Thomé]
5. Individual logarithm (a.k.a. "descent"): recursively rewrite "large" factors of h into products of smaller elements then reconstruct the corresponding DLs.
   [Detrey, Gaudry and Videau]

Table of Contents

DL Computation in $\mathrm{GF}(2^{809})$

Objective

Attack DLP in a subgroup of $\mathrm{GF}(2^{809})^\times$ of prime order ℓ, where ℓ is the 202-bit prime factor of $2^{809}-1$:

$$\ell = 4148386731260605647525186547488842396461625774241327567978137.$$

- $\mathrm{GF}(2^{809})^\times = p_{202} \times p_{607}$.
- This subgroup is large enough to resist to Pollard's ρ (101 bits of security).

- An equivalent of this computation using the new methods?

$\rightarrow$ DLP in $\mathrm{GF}(2^{809\times k})$, where $10 < k < 20$ (recall: record is $\mathrm{GF}(2^{9234})$).

DL Computation in $\mathrm{GF}(2^{809})$

Polynomial Selection

- For $f(x,t)$, the best choice was driven by Murphy's α value (quantity related to the efficiency of the relation collection):

$$f(x,t) = x^6 + \texttt{0x7}x^5 + \texttt{0x6b}x^3 + \texttt{0x1ab}x^2 + \texttt{0x326}x + \texttt{0x19b3}.$$

- For $g(x,t)$, no special care $\rightarrow$ monic linear polynomial with sparse constant term:

$$g(x,t) = x + \texttt{0x80000000000000000000000000000001e7eaa}.$$

- 2760 core-hours.
- Pre-computation phase, since f can be used to compute DLs in any field $\mathrm{GF}(2^n)$ with $700 \leq n \leq 900$.

A polynomial of $\mathrm{GF}(2)[\mathrm{t}]$ is represented by the value obtained when it is evaluated at $t = 2$, written in hexa. For instance, 0x7 represents $t^2 + t + 1$.

DL Computation in $\mathrm{GF}(2^{809})$

Relation Collection

Main parameters we play with:

- *Large-prime bound (B): limit for the degree of polynomials allowed in a relation. (a.k.a. the "smoothness bound")*
- *I,J*: dimensions of the sieved area.

2 sets of parameters tested:

B	I,J	degrees of special-$\mathfrak{q}$'s	#explored elts per sp.-$\mathfrak{q}$	#relations	CPU time (core-hours)
27	15	24 to 27	2^{30}	52M	37.2k
28	14	24 to 28	2^{28}	117M	26.9k

DL Computation in $\mathrm{GF}(2^{809})$

Filtering

3 stages:

1. **Duplicate**: remove duplicate relations.
2. **Purge**: remove singletons and relations while there are still more relations than ideals (i.e. more equations than unknowns).
3. **Merge**: beginning of Gaussian elimination.

B	27	28
#rels.	52M	117.4M
#uniq rels. (after duplicate)	30.1M	67.4M
#rels. after purge	9.6M	13.6M
final matrix (after merge)	3.7M	4.8M

DL Computation in $\mathrm{GF}(2^{809})$

Linear Algebra & Individual Logarithm

Linear algebra over $\mathbb{Z}/\ell\mathbb{Z}$: solve $Mw \equiv 0 \pmod{\ell}$

- M is sparse, ℓ is a 202-bit prime.
- Adapt a sparse format to represent M.
- Use of RNS representation to accelerate arithmetic over $\mathbb{Z}/\ell\mathbb{Z}$.
- Setup: 8 GPUs (NVIDIA Tesla M2050) on 4 nodes.
- Block Wiedemann ($m = 8$, $n = 4$): 4 sequences in parallel,
  1 sequence $\leftrightarrow$ 2 GPUs within the same CPU node.
- Wall-clock time: 4.5 days
- Overall time: 864 GPU-hours or 26.2k core-hours (CPU implem.)

Individual logarithm

- Classical descent by special-$\mathfrak{q}$.
- One individual log $\leq$ 1 h.

Table of Contents

Balancing Sieving and Linear Algebra

- For B=27, where to stop sieving?

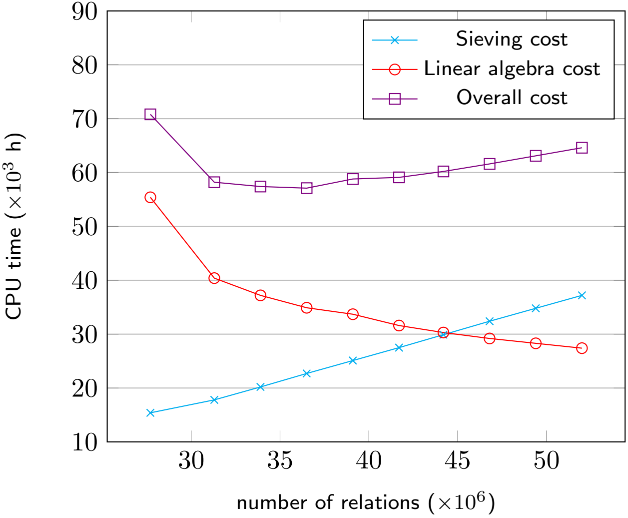

Table of Contents

Towards $\mathrm{GF}(2^{1039})$

Objective

Attack DLP in a subgroup of $\mathrm{GF}(2^{1039})^\times$ of prime order ℓ, where ℓ is the 265-bit prime factor of $2^{1039} - 1$.

Relation collection (*done*): 2.6 billion relations in 264 core-years.

Filtering (*done*): matrix of 60M rows and columns.

Linear algebra:

- GPUs cannot be used since RAM not sufficient (35 GB required).
- CPU implementation: 22 months (projected) on a 768-core cluster with Block Wiedemann ($m = 192$, $n = 96$).
- not yet launched:
  - try other parameters for sieving
  - feasibility of Block Wiedemann with these blocking parameters.

Conclusion

Assessment of the feasibility limit of DLs in $\mathrm{GF}(2^p)$ with FFS:

- DLP in $\mathrm{GF}(2^{809})^\times$ required 7.6 core-years and 0.1 GPU-years.
- DLP in $\mathrm{GF}(2^{1039})^\times$ is feasible with current hardware and software technology.

Investigation in steps used in the new algorithms:

- sieving
- linear algebra.

In the future:

- further experiments for FFS and for the new algorithms to establish the cut-off points between these algorithms for the prime degree extensions.

Unfortunately,

- One Nvidia GeForce GTX 680 (Gamer's card) burned out.

- The Ph.D thesis of *Nicolas Estibals* about the implementation of pairings in composite extension fields ruined due to $L(\frac{1}{4})$ and QPA.