Revisiting discrete logarithms in medium/small characteristic

Antoine Joux
UVSQ and CryptoExperts

August 15st, 2013

Discrete logarithms

- Given a multiplicative group G with generator g
- Computing discrete logarithms is inversing $n \to g^n$
- Hard in general and used as a hard problem in cryptography
- Algorithmic viewpoint
  - Generic algorithms (for any G)
  - Specific algorithms (make use of group representation)

Generic algorithms: Pohlig-Hellman

- Given a multiplicative group G with generator g
- Given $|G| = \prod_{i=1}^{k} p_i^{e_i}$
- To compute dlogs in G, it suffices to compute dlogs in:

$$G_i = \langle g^{|G|/p_i} \rangle \quad \text{(Group of order } p_i\text{)}$$

Generic algorithms: $|G| = p$

- There exist algorithms with complexity $O(\sqrt{p})$ to solve:

$$y = g^n$$

- Baby-step giant-step (let $R = \lceil \sqrt{p} \rceil$):
  - Create list $y, y/g, \cdots, y/g^{R-1}$
  - Create list $1, h, h^2, \cdots, h^{R-1}$, where $h = g^R$
  - Find collision
- Can be improved to memoryless algorithms using cycle finding techniques

Classical groups for Dlog in Cryptography

- Integers modulo p
- More general finite fields $\mathbb{F}_{p^k}$
- Elliptic curves over finite fields

Index calculus algorithms

- Relation generation phase
  - Generates many sparse equations
  - Modulo group order for discrete log
    (Modulo 2 for factoring)
- Linear algebra phase
  - Large sparse system
  - Numbers of unknowns in range up to dozens of millions
  - Number of equations potentially very large
  - Need to use large computers to solve such systems
- Individual logarithm phase

Index calculus: multiplicative generator ?

- After linear algebra:
  - Obtain an element of the kernel of equations
  - Modulo each (large) factor of group order
  - Discrete logarithms in any basis g_0 is a possible solution

- Conversely, **If**:
  - Matrix has "full" rank modulo each factor
  - And smoothness basis generates multiplicative group
- Then:
  - There exists g_0 such that kernel vector is Dlogs.
  - Moreover: any invertible entry in vector corresponds to a group generator.

- Alternative option that checks conditions:
  - Use different linear algebra (Smith normal form)
  - Proposed by Huang and Narayanan in *Finding Primitive Elements in Finite Fields of Small Characteristic.* [arXiv]

Complexity of Index calculus algorithms

- Write:

$$L_Q(\beta, c) = \exp((c + o(1))(\log Q)^{\beta}(\log\log Q)^{1-\beta}).$$

- Complexity of dlogs with index calculus algorithms
  - Number field sieve (p large):

$$L_p\left(1/3, \left(\frac{64}{9}\right)^{1/3}\right)$$

  - Number field sieve (p medium to large, $Q = p^k$):

$$L_Q\left(1/3, \left(\frac{128}{9}\right)^{1/3}\right)$$

  - Function field sieve (p small to medium, $Q = p^k$):

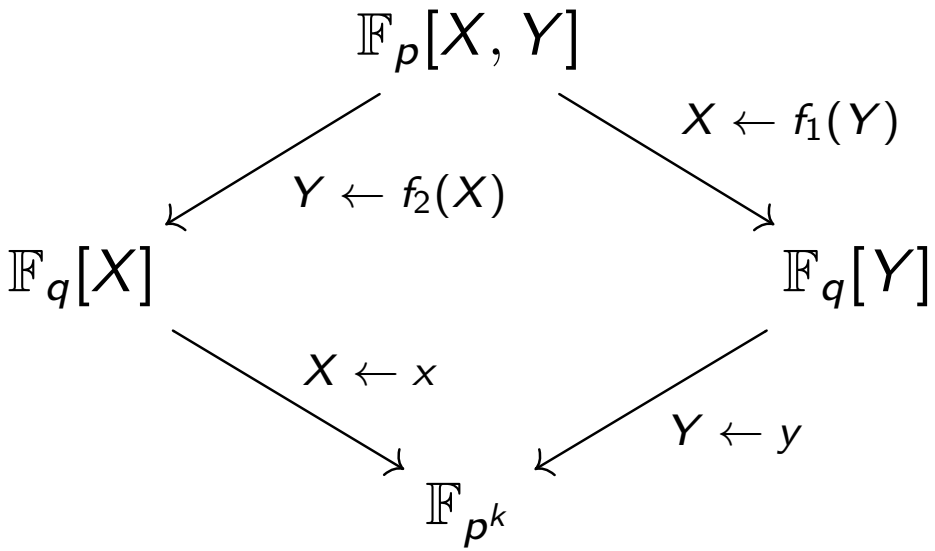
$$L_Q\left(1/3, \left(\frac{32}{9}\right)^{1/3}\right),$$

    the constant is reduced for some specific balance of p and k

Discrete Logarithms in the Medium prime case [JL06]

- Finite field of the form $\mathbb{F}_{p^k}$
- Choose two univariate polynomials f_1 and f_2
  - with degrees d_1 and d_2 and $d_1 d_2 \geq k$.
  - Such that $x - f_1(f_2(x))$ has:
    - an irreducible factor of degree k (modulo p).
- This defines the finite field by the relations:
  - $x = f_1(y)$ and $y = f_2(x)$

Commutative diagram



Discrete Logarithms in the Medium prime case [JL06]

- Optimal for $p = L_{p^k}(1/3)$

- Choose smoothness basis $x - \alpha$ and $y - \alpha$
- Consider elements:

$$\begin{aligned} xy + ay + bx + c &= x\, f_2(x) + a f_2(x) + bx + c \\ &= y\, f_1(y) + ay + b f_1(y) + c \end{aligned}$$

- When both sides split $\Rightarrow$ Relation
- Heuristic cost of finding relation (sieving):

$$(d_1 + 1)!\,(d_2 + 1)!$$

- Individual log. descent negligible compared to initial phase

Nice special case – Kummer extensions

- Assume $k|p-1$, then $\mathbb{F}_{p^k}$ can be defined by $x^k - t$
- If $k = d_1 d_2 - 1$, let $y = x^{d_1}$ and $tx = y^{d_2}$
- Reduces size of smoothness basis by k
  - Indeed:

$$\begin{aligned}(X+\alpha)^p &= X^p + \alpha = t^{(p-1)/k}X + \alpha = \mu(X + \alpha/\mu),\\ (Y+\alpha)^p &= \mu^{d_1}(Y + \alpha/\mu^{d_1}).\end{aligned}$$

    where μ is a k-th root of unity in $\mathbb{F}_p$.

- Can be generalized to $k = d_1 d_2 + 1$ using $y = x^{d_1}$ and $x = t/y^{d_2}$

Linear change of variables [J13]

- Further restrict to $y = x^{d_1}$
- Then:

$$xy + ay + bx + c \quad = \quad x^{d_1+1} + ax^{d_1} + bx + c$$

- Perform change of variable: $x = aX$, we get:

$$a^{d_1+1}(X^{d_1+1} + X^{d_1} + b \cdot a^{-d_1}(X + c/(ab)).$$

- Change of variable does not affect splitting property
- One good left-hand side $\Rightarrow$ p good left-hand sides
- Amortized cost of relation reduced to

$$\left(\frac{(d_1+1)!}{p-1} + 1\right) \cdot (d_2+1)!$$

Case of Kummer extensions

- Assume $k|p-1$, i.e. $\mathbb{F}_{p^k}$ can be defined by $x^k - t$
- If $k = d_1 d_2 - 1$, let $y = x^{d_1}$ and $tx = y^{d_2}$
  - $x^{d_1+1} + ax^{d_1} + bx + c \Rightarrow a^{d_1+1}(X^{d_1+1} + X^{d_1} + b \cdot a^{-d_1}(X + c/(ab))$.
  - $(y^{d_2+1} + by^{d_2})/t + ay + c \Rightarrow$
    $b^{d_2+1}\left((Y^{d_2+1} + Y^{d_2})/t + a \cdot b^{-d_2}(Y + c/(ab))\right)$.
- In both cases $\lambda = c/(ab)$ is shared by the two sides

Kummer extensions – Reassembling two sides

- Assume that:
  - $X^{d_1+1} + X^{d_1} + \theta_X(X + \lambda)$ splits and
  - $(Y^{d_2+1} + Y^{d_2})/t + \theta_Y(Y + \lambda)$ splits.
- Find a and b such that $\theta_X = b \cdot a^{-d_1}$ and $\theta_Y = a \cdot b^{-d_2}$?
- This implies $\theta_X^{d_2}\theta_Y = a^{-d_1 d_2+1} = a^{-k}$.
  - Possible iff $\theta_X^{d_2}\theta_Y$ is a k-th power
  - Gives k (conjugate) solutions !
  - From a recover b and c
  - Roots obtained by change of variable

Impact in the medium prime case

- In theory, reduces constant in $L(1/3)$ complexity of function field sieve.

- In practice, Kummer extensions esp. good for records:
  - First 1175-bit field $\mathbb{F}_{p^{47}}$ with p close to 2^{25}
  - Then 1425-bit field $\mathbb{F}_{p^{57}}$ with p close to 2^{25}
  - Previous finite field record was 923 bits
  - Timings: about 32000 CPU-hours compared to 895000 CPU-hours

- $47 = 6 \cdot 8 - 1$
- $57 = 7 \cdot 8 + 1$

Small characteristic – Setting [J13b]

- Define finite field by a relation:

$$x^{p^\ell} = \frac{h_0(x)}{h_1(x)},$$

gives degree $k = \deg(I(x))$ extension, where $I(x)$ is a divisor of $h_1(x)x^{p^\ell} - h_0(x)$.

- We have a systematic relation:

$$x^{p^\ell} - x = \prod_{\alpha \in \mathbb{F}_{p^\ell}} (x - \alpha).$$

Small characteristic – Basic idea [J13b]

- Use more general change of variable: $x = \frac{aX+b}{cX+d}$, we get:

$$(cX+d)\cdot(aX+b)^{p^\ell} - (aX+b)\cdot(cX+d)^{p^\ell} =$$
$$(cX+d)\cdot \prod_{\alpha\in\mathbb{F}_{p^\ell}} ((a-\alpha\, c)X + (b-\alpha\, d)$$

- Moreover, after expanding the left-hand size, we find:

$$(ca^q - ac^q)X^{q+1} + (da^q - bc^q)X^q + (cb^q - ad^q)X + (db^q - bd^q),$$

  where $q = p^\ell$.
  It becomes a low degree polynomial after multiplying by h_1 and replacing $h_1(X)\, X^q$.
- As a consequence, multiplicative relations are very easy to find

Small characteristic – Choice of a, b, c and d

- If a, b, c and d are in $\mathbb{F}_q$ left-hand side is:

$$(ad - bc)(X^q - X) \Rightarrow \text{Trivial relation}$$

- Take a, b, c and d in small extension field such as $\mathbb{F}_{q^2}$
- Some choices of (a, b, c, d) are equivalent. Good parametrization is:

$$PGL_2(\mathbb{F}_{q^2})/PGL_2(\mathbb{F}_q)$$

Small characteristic – Resulting Complexity [J13b]

- Logarithms of smoothness basis in polynomials time
  - Because base field is very small compared to extension field
- Hard part is individual logarithms
  - Usual descent algorithm not good enough
  - Need to be completed by new descent algorithm (based on resolution of bilinear systems of Equations)
  - Resulting complexity is:

$$L(1/4 + o(1)).$$

  - Practical application:
    - New records in $\mathbb{F}_{2^{1778}}$, $\mathbb{F}_{2^{4080}}$ and $\mathbb{F}_{2^{6168}}$ recently announced
    - Other records by Göloğlu, Granger, McGuire and Zumbrägel

- $1778 = 2 \cdot 7 \cdot (2^7 - 1)$, 220 CPU-hours
- $4080 = 2 \cdot 8 \cdot (2^8 - 1)$, 14100 CPU-hours
- $6168 = 3 \cdot 8 \cdot (2^8 + 1)$ 550 CPU-hours

Descent phase (option 1)

- In practice, bootstrap using continued fractions
- Classical descent (for high to mid degrees).
- New descent (for mid to low degrees), based on a system a bilinear equations.

Descent phase (option 2)

- Joint work with Barbulescu, Gaudry, Thomé
- Without Gröbner bases
- Improved complexity:

$$\exp(O(\log q \log k)).$$

- Sub-exponential but not practical (yet)
- Basic idea, evaluate $X^q - X$ at homography in $P(x)$ and 1

Conclusion

Questions ?