# Computing Discrete Logarithms using Joux's Algorithm

Gora Adj, Thomaz Oliveira, Francisco Rodríguez-Henríquez
CINVESTAV-IPN (Mexico)

Alfred Menezes
University of Waterloo (Canada)

gora.adj@gmail.com

**Abstract**

In 2013, Joux presented a new algorithm for solving the discrete logarithm problem in finite fields of small characteristic with a main novelty involving the resolution of bilinear equation systems. The algorithm improved significantly all previous methods for this purpose. We used Joux's algorithm to compute discrete logarithms in the 1303-bit finite field $\mathbb{F}_{3^{6\cdot 137}}$ and illustrated for the first time its effectiveness in 'general' small-characteristic finite fields with very modest computational resources.

**Keywords**
Discrete logarithm problem, Joux's algorithm, Gröbner bases descent.

## 1 Introduction

Let $\mathbb{F}_Q$ denote the finite field of order $Q$. The discrete logarithm problem (DLP) in $\mathbb{F}_Q$ is that of determining, given a generator $g$ of $\mathbb{F}_Q^*$ and an element $h \in \mathbb{F}_Q^*$, the integer $x = \log_g h \in [0, Q-2]$ satisfying $h = g^x$. Hereafter we suppose that the characteristic of $\mathbb{F}_Q$ is 2 or 3.

Until recently, the fastest general-purpose algorithm known for solving the DLP in $\mathbb{F}_Q$ was Coppersmith's 1984 index-calculus algorithm [4] with a running time of $L_Q[\frac{1}{3}, (32/9)^{1/3}] \approx L_Q[\frac{1}{3}, 1.526]$, where as usual $L_Q[\alpha, c]$ with $0 < \alpha < 1$ and $c > 0$ denotes the expression

$$\exp\left((c + o(1))(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$$

that is subexponential in $\log Q$. In February 2013, Joux [12] presented a new DLP algorithm with a running time of $L_Q[\frac{1}{4} + o(1), c]$ (for some undetermined $c$) when $Q = q^{dn}$, $q \approx n$ and $d$ a small integer. Shortly thereafter, Barbulescu, Gaudry, Joux and Thomé [3] presented an algorithm with *quasi-polynomial* running time $(\log Q)^{O(\log\log Q)}$ when $Q = q^{dn}$ with $q \approx n$ and $d$ a small integer. Note that this complexity is asymptotically smaller than $L_Q[\alpha, c]$ for any $\alpha > 0$ and $c > 0$.

After a concrete analysis of the new algorithms, we demonstrated in [1] that these two algorithms can be combined to weaken some large fields that were originally believed crytographically secure (against Coppersmith algorithm).

Recently, we employed, for the first time, Joux's algorithm to compute discrete logarithms in a 'general' cryptographic field, namely $\mathbb{F}_{3^{6\cdot 137}}$ (also in $\mathbb{F}_{3^{6\cdot 163}}$ [2]), which does not enjoy any specific properties. The computations of a discrete logarithm took only 888 CPU hour using modest computer resources despite our implementation being in Magma [14] and far from optimal. By comparison, in 2012 Hayashi et al. [11] used the Joux-Lercier algorithm [13] (of complexity $L[1/3]$) to compute a discrete logarithm in $\mathbb{F}_{3^{6\cdot 97}}$ in about 896313 CPU hours.

In Joux's $L[1/4]$ index-calculus algorithm, the Gröbner bases descent phase, where bilinear equation systems need to be solved, constitutes a crucial step for the computation of a discrete logarithm in finite fields of small characteristic. In fact, the complexity of the algorithm is given by that of the Gröbner bases descent, based on the work previously done by Faugère, Safey El Din and Spaenlehauer [7].

In §2, we describe Joux's algorithm with a focus on the Gröbner bases descent. Next, we discuss our experimental results in §3 with computing a discrete logarithm in $\mathbb{F}_{3^{6\cdot 137}}$, and draw our conclusions in §4.

# 2 The index-calculus algorithm of Joux

Let $\mathbb{F}_{q^{dn}}$ be a finite field where $n \le 2q+1$ and $d > 1$ a small integer. The elements of $\mathbb{F}_{q^{dn}}$ are represented as polynomials of degree at most $n-1$ over $\mathbb{F}_{q^d}$. Let $N = q^{dn}-1$. Let $g$ be an element of order $N$ in $\mathbb{F}_{q^{dn}}^*$, and let $h \in \mathbb{F}_{q^{dn}}^*$. We wish to compute $\log_g h$. The algorithm proceeds by first finding the logarithms of all degree-one (and degree-two elements whenever $d = 2$) in $\mathbb{F}_{q^{dn}}$. Then, after the *descent stage*, $\log_g h$ is expressed as a linear combination of logarithms of degree-one (and degree-two if $d = 2$) $\mathbb{F}_{q^{dn}}$ elements. The descent stage proceeds in several steps, each expressing the logarithm of a degree-$D$ element as a linear combination of the logarithms of elements of degree $\le m$ for some $m < D$. The final step in the descent tree is dedicated to small degrees $D$ and requires to solve a bilinear system which can be done by computing a Gröbner basis.

The representation of the targeted fields is given in §2.1, and in §2.2 the Gröbner bases descent is detailed.

## 2.1 Setup

Select polynomials $h_0, h_1 \in \mathbb{F}_{q^d}[X]$ of degree at most 2 so that the polynomial of Granger et al. [10]

$$X \cdot h_1(X^q) - h_0(X^q) \tag{1}$$

has an irreducible factor $I_X$ of degree $n$ in $\mathbb{F}_{q^d}[X]$; we will henceforth assume that $\max(\deg h_0, \deg h_1) = 2$, whence $n \le 2q+1$. Note that

$$X \;\equiv\; \frac{h_0(X^q)}{h_1(X^q)} \;\equiv\; \left(\frac{\overline{h}_0(X)}{\overline{h}_1(X)}\right)^q \quad (\text{mod } I_X), \tag{2}$$

where for $P \in \mathbb{F}_{q^d}[X]$, $\overline{P}$ denotes the polynomial obtained by raising each coefficient of $P$ to the power $q^{d-1}$. The field $\mathbb{F}_{q^{dn}}$ is represented as $\mathbb{F}_{q^{dn}} = \mathbb{F}_{q^d}[X]/(I_X)$ and the elements of $\mathbb{F}_{q^{dn}}$ are represented as polynomials in $\mathbb{F}_{q^d}[X]$ of degree at most $n-1$. Let $g$ be a generator of $\mathbb{F}_{q^{dn}}^*$.

## 2.2 Gröbner bases descent

Let $f \in \mathbb{F}_{q^d}[X]$ with $\deg f = D$. Let $m = \lceil (D+1)/2 \rceil$ (or $m = 1$ when $d > 2$ and $D = 2$). Suppose that $\log_g \overline{h}_1$ is known and $3m < n$. In Joux's new descent method [12, §5.3], one finds degree-$m$ polynomials[1] $k_1, k_2 \in \mathbb{F}_{q^d}[X]$ such that $f \mid G$, where

$$G^q = (\overline{h}_1)^{mq}(k_1^q k_2 - k_1 k_2^q) \bmod I_X.$$

We then have

$$(\overline{h}_1)^{mq} \cdot k_2 \cdot \prod_{\alpha \in \mathbb{F}_q} (k_1 - \alpha k_2) \equiv G^q \pmod{I_X}$$

as can be seen by making the substitution $Y \mapsto k_1/k_2$ into the systematic equation

$$Y^q - Y = \prod_{\alpha \in \mathbb{F}_q} (Y - \alpha), \tag{3}$$

and clearing denominators. Now, if $k(X) = a_m X^m + a_{m-1}X^{m-1} + \cdots + a_0 \in \mathbb{F}_{q^d}[X]$, then

$$\begin{aligned} k(X) \;&\equiv\; a_m \left(\left(\frac{\overline{h}_0(X)}{\overline{h}_1(X)}\right)^q\right)^m + a_{m-1}\left(\left(\frac{\overline{h}_0(X)}{\overline{h}_1(X)}\right)^q\right)^{m-1} + \cdots + a_0 \quad (\text{mod } I_X) \\ &\equiv\; \left(\overline{a}_m \left(\frac{\overline{h}_0(X)}{\overline{h}_1(X)}\right)^m + \overline{a}_{m-1}\left(\frac{\overline{h}_0(X)}{\overline{h}_1(X)}\right)^{m-1} + \cdots + \overline{a}_0\right)^q \quad (\text{mod } I_X), \end{aligned}$$

where for $\gamma \in \mathbb{F}_{q^d}$, $\overline{\gamma}$ denotes the element $\gamma^{q^{d-1}}$. Whence after clearing denominators

$$k(\overline{h}_1)^{mq} \equiv \left(\overline{a}_m(\overline{h}_0)^m + \overline{a}_{m-1}(\overline{h}_0)^{m-1}(\overline{h}_1) + \cdots + \overline{a}_0(\overline{h}_1)^m\right)^q \pmod{I_X}.$$

[1] More generally, the degrees of $k_1$ and $k_2$ can be different.

Define $\widetilde{k} = k(\overline{h}_1)^{mq}$, and note that $\deg \widetilde{k} = 2m$. We thus have $G \equiv k_1\widetilde{k}_2 - \widetilde{k}_1 k_2 \pmod{I_X}$, and consequently $G = k_1\widetilde{k}_2 - \widetilde{k}_1 k_2$ since $3m < n$. It follows that $G(X) = f(X)R(X)$ for some $R \in \mathbb{F}_{q^d}[X]$ with $\deg R = 3m - D$. If $R$ is $m$-smooth, we obtain a linear relationship between $\log_g f$ and logarithms of degree-$m$ polynomials by taking logarithms of both sides of the following:

$$(\overline{h}_1)^{mq} \cdot k_2 \cdot \prod_{\alpha \in \mathbb{F}_q} (k_1 - \alpha k_2) \equiv f(X)^q R(X)^q \pmod{I_X}. \tag{4}$$

To determine $(k_1, k_2, R)$ that satisfy

$$k_1\widetilde{k}_2 - \widetilde{k}_1 k_2 = fR, \tag{5}$$

one can transform (5) into a system of multivariate bilinear equations over $\mathbb{F}_q$. Specifically, each coefficient of $k_1$ and $k_2$ is written using $d$ variables over $\mathbb{F}_q$, the $d$ variables representing the $d$ components of that coefficient (which is in $\mathbb{F}_{q^d}$) over $\mathbb{F}_q$. The coefficients of $\widetilde{k}_1$, $\widetilde{k}_2$ and $R$ can be written in terms of the coefficients of $k_1$ and $k_2$ and $f$. Hence, equating coefficients of $X^i$ of both sides of (5) yields $3m+1$ quadratic equations. The $\mathbb{F}_q$-components of each of these equations are equated, yielding $dD$ bilinear equations in $2d(m+1)$ variables over $\mathbb{F}_q$. This system of equations is then solved by finding a Gröbner basis for the ideal it generates. Finally, solutions $(k_1, k_2, R)$ are tested until one is found for which $R$ is $m$-smooth. This yields an expression for $\log_g f$ in terms of the logarithms of at most $(q+1) + (3m - D)$ polynomials of degree at most $m$.

Now, considering the action of $\mathrm{Gl}_2(\mathbb{F}_{q^d})$ on the set of pairs $(k_1, k_2)$, one expects to have enough relations for descending any irreducible polynomial over $\mathbb{F}_{q^d}[X]$ of degree $D$ whenever

$$q^{(2m+1-D)d-3} > p^{-1}, \tag{6}$$

where $p$ is the probability of $R$ to be $m$-smooth [9, Section 3.3].

# 3 Computing discrete logarithms in $\mathbb{F}_{3^{6\cdot 137}}$

The supersingular elliptic curve $E : y^2 = x^3 - x + 1$ has order $\#E(\mathbb{F}_{3^{137}}) = cr$, where

$$c = 7 \cdot 4111 \cdot 5729341 \cdot 42526171$$

and

$$r = (3^{137} - 3^{69} + 1)/c = 33098280119090191028775580055082175056428495623$$

is a 155-bit prime. The Weil and Tate pairing attacks [15, 8] efficiently reduce the logarithm problem in the order-$r$ subgroup $\mathcal{E}$ of $E(\mathbb{F}_{3^{137}})$ to the discrete logarithm problem in the order-$r$ subgroup $\mathcal{G}$ of $\mathbb{F}^*_{3^{6\cdot 137}}$.

Our approach to compute logarithms in $\mathcal{G}$ is to use Joux's algorithm to compute logarithms in the quadratic extension $\mathbb{F}_{3^{12\cdot 137}}$ of $\mathbb{F}_{3^{6\cdot 137}}$ (so $q = 3^4$, $n = 137$ and $d = 3$ in the notation of §2). More precisely, we are given two elements $\alpha, \beta$ of order $r$ in $\mathbb{F}^*_{3^{12\cdot 137}}$ and we wish to find $\log_\alpha \beta$. Let $g$ be a generator of $\mathbb{F}^*_{3^{12\cdot 137}}$. Then $\log_\alpha \beta = (\log_g \beta)/(\log_g \alpha) \bmod r$. Thus, in what follows we will assume that we need to compute $\log_g h \bmod r$, where $h$ is an element of order $r$ in $\mathbb{F}^*_{3^{12\cdot 127}}$.

Our DLP instance is described in §3.1 and the experimental results are presented in §3.2.

## 3.1 Problem instance

Let $N$ denote the order of $\mathbb{F}^*_{3^{12\cdot 137}}$. Using the tables from the Cunningham Project [5], we determined that the factorization of $N$ is $N = p_1^4 \cdot \prod_{i=2}^{31} p_i$, where the $p_i$ are all known primes (and $r = p_{25}$).

We chose the representations $\mathbb{F}_{3^4} = \mathbb{F}_3[U]/(U^4 + U^2 + 2)$ and $\mathbb{F}_{3^{12}} = \mathbb{F}_{3^4}[V]/(V^3 + V + U^2 + U)$, and selected

$$h_0(X) = V^{326196}X^2 + V^{35305}X + V^{204091} \in \mathbb{F}_{3^{12}}[X]$$

and $h_1 = 1$. Then $I_X \in \mathbb{F}_{3^{12}}[X]$ is the degree-137 monic irreducible factor of $X - h_0(X^{3^4})$; the other irreducible factor has degree 25.

We chose the generator $g = X + V^{113713}$ of $\mathbb{F}^*_{3^{12\cdot 137}}$. To generate an order-$r$ discrete logarithm challenge $h$, we computed

$$h' = \sum_{i=0}^{136} \left( V^{\lfloor \pi \cdot (3^{12})^{i+1} \rfloor \bmod 3^{12}} \right) X^i$$

and then set $h = (h')^{N/r}$. The discrete logarithm $\log_g h \bmod r$ was found to be

$$x = 27339619076975093920245515973214186963025656559.$$

This can be verified by checking that $h = (g^{N/r})^y$, where $y = x \cdot (N/r)^{-1} \bmod r$ (cf. Appendix A).

### 3.2 Experimental results

Our experiments were run on an Intel i7-2600K 3.40 GHz machine (Sandy Bridge), and on an Intel i7-4700MQ 2.40 GHz machine (Haswell).

The number of degree-1 elements of $\mathbb{F}_{q^{dn}}$ that we had to compute their logarithms is $3^{12} \approx 2^{19}$. After finding enough linear relations ($2^{19}$ therefore) between the logarithms of these elements in 1.05 CPU hours (Sandy Bridge, 1 core), the resulting sparse linear system of linear equation was solved modulo $r$ using Magma's multi-threaded parallel version of the Lanczos algorithm in 556.8 CPU hours (Sandy Bridge, 4 cores). Since in this case $d > 2$, we did not need to compute the logarithms of all the degree-2 elements of $\mathbb{F}_{q^{dn}}$.

The first two classical methods in the descent phase allowed to express the logarithm of our challenge element (of degree 136) as a linear combination of logarithms of polynomials of degree $\leq 5$ in 102 CPU hours (Haswell, 4 cores).

The Magma implementation of Faugère's F4 algorithm [6] is utilized for the Gröbner bases descent stage, and this takes 26.5 minutes on average for a degree-5 to degree-3 descent, 33.8 seconds for a degree-4 to degree-3 descent, 34.7 seconds for a degree-3 to degree-2 descent, and 0.216 seconds for a degree-2 to degree-1 descent. In total, we performed 233 5-to-3 descents, 174 4-to-3 descents, and 11573 3-to-2 descents. These computations took 115.2 CPU hours, 1.5 CPU hours, and 111.2 CPU hours, respectively (Haswell, 4 cores). We also performed 493537 2-to-1 descents; their running times are incorporated into the running times for the higher-level descents.

## 4 Conclusions

We discussed Joux's $L[1/4]$ DLP algorithm, focusing on the Gröbner bases descent stage, and used the algorithm to solve a problem instance in the 1303-bit finite field $\mathbb{F}_{3^{6 \cdot 137}}$ in 888 CPU hours. However, it remains to be determined if the DLP in higher extensions (for example, in the 2273-bit field $\mathbb{F}_{3^{6 \cdot 239}}$) can be solved using modest computational resources. Our preliminary analysis on the field $\mathbb{F}_{3^{6 \cdot 239}}$ is pessimistic. A main obstacle is the cost of finding a Gröbner basis of a bilinear system, which grows very rapidly with the number of the variables when one wishes to increase the parameter $d$ or the degree of the field representation polynomials $h_0$ and $h_1$.

## References

[1] G. Adj, A. Menezes, T. Oliveira and F. Rodríguez-Henríquez, "Weakness of $\mathbb{F}_{3^{6 \cdot 509}}$ for discrete logarithm cryptography", *Pairing-Based Cryptography – Pairing 2013*, LNCS 8365 (2014), 20–44.

[2] G. Adj, A. Menezes, T. Oliveira and F. Rodríguez-Henríquez, "Computing discrete logarithms in $\mathbb{F}_{3^{6 \cdot 137}}$ and $\mathbb{F}_{3^{6 \cdot 163}}$ using Magma", available at `http://eprint.iacr.org/2014/057`.

[3] R. Barbulescu, P. Gaudry, A. Joux and E. Thomé, "A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic: Improvements over FFS in small to medium characteristic", *Advances in Cryptology – EUROCRYPT 2014*, LNCS 8441 (2014), 1–16.

[4] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two", *IEEE Transactions on Information Theory*, 30 (1984), 587–594.

[5] The Cunningham Project, `http://homes.cerias.purdue.edu/ ssw/cun/`.

[6] J.-C. Faugère, "A new efficient algorithm for computing Gröbner bases (F4)", *Journal of Pure and Applied Algebra*, 139(1-3) (1999), 61–88.

[7] J.-C. Faugère, M. Safey El Din and P.-J. Spaenlehauer, "Gröbner Bases of Bihomogeneous Ideals Generated by Polynomials of Bidegree (1,1): Algorithms and Complexity", *Journal of Symbolic Computation*, 46(4) (2011), 406–437.

[8] G. Frey and H. Rück, "A remark concerning $m$-divisibility and the discrete logarithm in the divisor class group of curves", *Mathematics of Computation*, 62 (1994), 865–874.

[9] R. Granger, T. Kleinjung and J. Zumbrägel, “Breaking ‘128-bit secure’ supersingular binary curves (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$)”, *Advances in Cryptology – CRYPTO 2014*, to appear; available at http://eprint.iacr.org/2014/119.

[10] R. Granger and J. Zumbrägel, “On the security of supersingular binary curves”, presentation at ECC 2013, September 16 2013.

[11] T. Hayashi, T. Shimoyama, N. Shinohara and T. Takagi, “Breaking pairing-based cryptosystems using $\eta_T$ pairing over $GF(3^{97})$”, *Advances in Cryptology – ASIACRYPT 2012*, LNCS 7658 (2012), 43–60.

[12] A. Joux, “A new index-calculus algorithm with complexity $L(1/4+o(1))$ in very small characteristic”, *Selected Areas in Cryptography – SAC 2013*, LNCS 8282 (2014), 355–379.

[13] A. Joux and R. Lercier, “The function field sieve in the medium prime case” *Advances in Cryptology – EUROCRYPT 2006*, LNCS 4004 (2006), 254–270.

[14] Magma v2.19-7, http://magma.maths.usyd.edu.au/magma/.

[15] A. Menezes, T. Okamoto and S. Vanstone, “Reducing elliptic curve logarithms to logarithms in a finite field”, *IEEE Transactions on Information Theory*, 39 (1993), 1639–1646.

# A Magma script for verifying the $\mathbb{F}_{3^{6\cdot 137}}$ discrete logarithm

```
//Definition of the extension fields Fq := F3(U) and Fq3 := Fq(V)
q        := 3^4;
F3       := FiniteField(3);
P3<u>    := PolynomialRing(F3);
poly     := u^4 + u^2 + 2;
Fq<U>    := ext<F3|poly>;
Pq<v>    := PolynomialRing(Fq);
poly     := v^3 + v + U^2 + U;
Fq3<V>   := ext<Fq|poly>;
Pq3<Z>   := PolynomialRing(Fq3);
r        := 33098280119090191028775580055082175056428495623;
Fr       := GF(r);

h0       := V^326196*Z^2 + V^35305*Z + V^204091;
h0q      := Evaluate(h0,Z^q);
F        := Z - h0q;
Ix       := Factorization(F)[2][1];
Fn<X>    := ext<Fq3|Ix>;
N        := #Fn - 1;

// Generator of GF(3^{12*137})^*
g        := X + V^113713;

// Encoding pi
Re       := RealField(2000);
pival    :=Pi(Re);
hp       := 0;
for i := 0 to 136 do
         hp := hp + V^(Floor(pival*(#Fq3)^(i+1)) mod #Fq3)*(X^i);
end for;

// This is the logarithm challenge
cofactor := N div r;
h        := hp^cofactor;

// log_g(h) mod r is:
x        := 27339619076975093920245515973214186963025656559;

// Define the exponent y to be used in the verification:
y        := IntegerRing()!(Fr!(x/cofactor));

// Check that h = (g^cofactor)^y
h eq (g^cofactor)^y;
```