A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic (Draft)

Antoine Joux

CryptoExperts and
Université de Versailles Saint-Quentin-en-Yvelines, Laboratoire PRISM,
45 avenue des États-Unis, F-78035 Versailles Cedex, France
antoine.joux@m4x.org

Abstract. In this paper, we describe a new algorithm for discrete logarithms in small characteristic. It works especially well when the characteristic is fixed. Indeed, in this case, we obtain a total complexity of $L(1/4 + o(1))$.

1 Introduction

The discrete logarithm problem is one of the major hard problems used in cryptography. In this paper, we show that in small characteristic, it can be solved with a smaller heuristic complexity than previously believed.

In the course of writing this paper, that in particular explains the computation announced in [4], we discovered that another team has been working on similar ideas [1] and used them to compute discrete logarithms [2] in $\mathbb{F}_{2^{1971}}$.

An important distinction between the two works is their range of applicability. The authors of [1] consider extensions of the form $\mathbb{F}_{q^n}$ with

$$n = \frac{1}{\alpha} \cdot \left(\frac{\log Q}{\log\log Q}\right)^{2/3}, \qquad q = \exp\left(\alpha \cdot \sqrt[3]{\log Q \cdot \log^2 \log Q}\right),$$

where $Q = q^n$.

In this paper, we look at extensions $\mathbb{F}_{q^{2k}}$, where $q \approx k$, i.e., we consider much smaller base fields.

In the present competitive context, timeliness is preferable to completeness. We have thus chosen to publish this preprint in a draft form that includes neither a complete analysis of our results, nor a large number of examples. We apologize to the reader for any inconvenience.

1.1 Refresher on function field sieve algorithms

When considering the computation of discrete logarithm in fields of the form $\mathbb{F}_{q^n}$, where q is relatively small compared to q^n, the state of the art choice is to use one of the numerous variation of the function field sieve. For larger values of q, it becomes preferable to use a variation of the number field sieve. The choice between the two algorithms is made by comparing q and $L_{q^n}(\frac{1}{3})$.

Both the function field sieve and the number field sieve algorithm are index calculus algorithms and can be decomposed in three main phases. After a preliminary step that chooses the representation of the field to be addressed and a smoothness basis, they first run a relation collection phase which produces many multiplicative relations between

elements of the smoothness basis. The second phase solves the linear system derived from the coefficient of the multiplicative relations and produces a vector of consistent logarithms for all elements in the smoothness basis. The final phase receives as input an arbitrary elememt of the finite field and expresses it as a product of (powers of) elements of the smoothness basis. This allows us to compute the discrete logarithm of this element.

Despite this common structure, index calculus algorithms may greatly differ in the way they produce their multiplicative relations. Since in this paper, we only consider small values of q, let us consider the production of relations in function field sieve algorithms. Essentially, relations are produced by considering an finite field element which can be represented in two distinct ways by an element of a function field above $\mathbb{F}_q$ (or simply of a ring of polynomials). When both representations can be factored into elements of the smoothness basis, this leads to a multiplicative relation.

To test whether a function field element can be factored, one takes its norm, which is a polynomial with coefficient in $\mathbb{F}_q$, and tests whether it can be factored into a product of low-degree polynomials. As the consequence, the keystone of function field sieve algorithms is our understanding of the factorization of two related polynomials into low-degree factors. In general, we don't know much about this. However, practical experiments gave good indication that even related polynomial often behaves as random polynomial in this respect. In fact, current function sieve algorithm rely on this as an essential heuristic assumption. A notable exception is the algorithm described in [1], which independently considered a similar approach.

The first deviation from the classical smoothness approach appeared in [3]. This paper remarked that, using a linear change of variable, a single polynomial with good factorization is amplified into many. Thanks to specific choice of parameters, this can be done while keeping the ability to find a second representation of the same elements. Of course, it remains essential that this second representation also has a good probability of factoring into low-degree polynomials.

Another alternative is also proposed in [3]: taking two distinct representations with a low-degree factorization and uses a linear change on each of them to coerce them into two distinct representations of a common element. However, the construction of the initial polynomials still involves a random search for a polynomial that nicely factors.

1.2 Notations

As usual when studying index calculus algorithms, we write:

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

As classical useful result is the logarithm of the probability that a **random** polynomial of degree n decomposes into factors of degree m over a finite field is close to:

$$-\frac{n}{m}\log\left(\frac{n}{m}\right),$$

for a wide range of parameters [6].

In index calculus algorithm, a standard heuristic assumption is to assume that all polynomials that arise in the algorithm also follow this smoothness probability. In our algorithm, this is false by construction, because we consider polynomials than decompose more frequently than usual. However, we still use the heuristic assumption on some polynomials, for which we have no known reason to say that they deviate from the *normal* behavior.

1.3 Recalling the algorithm from [5]

The medium prime discrete logarithms proposed in [5] works as follows. In order to compute discrete logarithms in $\mathbb{F}_{q^n}$, a degree n extension of the base field $\mathbb{F}_q$, it starts by defining the extension field implicitly from two bivariate polynomials in X and Y:

$$f_1(X,Y) = X - g_1(Y), \qquad f_2(X,Y) = -g_2(X) + Y,$$

where g_1 and g_2 are univariate polynomials of degree d_1 and d_2. In order to define the expected extension, this requires that the polynomial $-g_2(g_1(Y)) + Y$ has an irreducible factor $F(Y)$ of degree n over $\mathbb{F}_q$. As explained in [5], it is easy to find polynomials g_1 and g_2 that satisfy this requirement.

The relative degrees of d_1 and d_2 in this case are controlled by an extra parameter D, whose choice is determined by the size of q compared to q^n. More precisely, we have $d_1 \approx \sqrt{Dn}$ and $d_2 \approx \sqrt{n/D}$.

Starting from this definition of the finite field, the medium prime field algorithms consider objects of the form $\mathcal{A}(Y)\,X + \mathcal{B}(Y)$, where $\mathcal{A}$ and $\mathcal{B}$ are univariate polynomials of degree D and $\mathcal{A}$ is unitary. Substituting $g_1(Y)$ for X on one side and $g_2(X)$ for Y on the other, we obtain an equation:

$$\mathcal{A}(Y)\,g_1(Y) + \mathcal{B}(Y) = \mathcal{A}(g_2(X))\,X + \mathcal{B}(g_2(X)).$$

This relates a polynomial of degree $d_1 + D$ in Y and a polynomial of degree $Dd_2 + 1$ in X.

To use the equations as index calculus relations, the algorithm of [5] selects the set of all unitary polynomials of degree at most D in X or Y, with coefficients in $\mathbb{F}_q$ as its smoothness basis and keeps pairs of polynomials (a, b) such that the two polynomials $a(Y)\,g_1(Y) + b(Y)$ and $a(g_2(X))\,X + b(g_2(X))$ both factor into terms of degree at most D. These good pairs are found using a classical sieving approach.

Writing $Q = q^n$, to analyze the complexity of the medium prime discrete logarithms, [5] chooses to write $q = L_Q(\frac{1}{3}, \alpha\,D)$, where as usual:

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

In this setting, the (heuristic) asymptotic complexity of the sieving phase is $L_Q(\frac{1}{3}, c_1)$ and the complexity of the linear algebra is $L_Q(\frac{1}{3}, c_2)$, with:

$$c_1 = \frac{2}{3\sqrt{\alpha D}} + \alpha D \quad \text{and} \quad c_2 = 2\alpha D.$$

Note that the algorithm with parameter D only works under the condition:

$$(D+1)\alpha \geq \frac{2}{3\sqrt{\alpha D}}. \tag{1}$$

Otherwise, the number of expected relations is too small to relate all elements of the smoothness basis. For a finite field $\mathbb{F}_{q^n}$, [5] indicates that the best complexity is obtained choosing the smallest acceptable value for the parameter D.

Individual discrete logarithms phase Another very important phase that appears in many index calculus based algorithms is the individual discrete logarithms phase which allows to compute the logarithm of an arbitrary field element by finding a multiplicative relation which relates this element to the elements of the smoothness basis whose logarithms have already been computed.

In [5], this is done by first expressing the desired element as a product of elements which can be represented as low degree polynomials in X or Y. These polynomials can in turn be related to polynomials of a lower degree and so on, until hitting degree one, i.e. elements of the smoothness basis. For this reason, the individual logarithm phase is also called the descent phase.

As analyzed in [5], the asymptotic complexity of the descent phase is

$$L_Q\left(\frac{1}{3}, \frac{1}{3\mu\sqrt{\alpha D}}\right),$$

where $\mu < 1$ is an arbitrary parameter. Moreover, any choice of μ in the interval $\left]\frac{1}{2}; 1\right[$ ensures that the complexity of the descent phase is asymptotically negligible compared to (at least one of) the main phases.

2 New algorithm: the basic ideas

The new index calculus algorithms proposed in this paper hinges on two basic ideas.

Basic idea 1. In [3], it was remarked that a polynomial f that nicely factors can be transformed into several such polynomials, simply by a linear change of variable: $f(x) \longrightarrow f(ax)$, for any constant a.

Our first idea consists in remarking that this is also true for a larger class of change of variables. Basically, we consider changes given homographies:

$$x \longrightarrow \frac{ax+b}{cx+d}.$$

The reader might object that an homography is not going to transform f into polynomial. To cover this, we instead perform homogeneous evaluation of f at $(ax+b)/(cx+d)$.

In other words, we consider the polynomial:

$$F_{abcd}(x) = (cx+d)^{\deg f} f\left(\frac{ax+b}{cx+d}\right).$$

Theorem 1. *Let $f(Y)$ be a monic polynomial of degree D over $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$ be an extension field of $\mathbb{F}_q$. Let $F_{abcd}(U) = (cx+d)^{\deg f} f\left(\frac{ax+b}{cx+d}\right)$ with $(a,b,c,d) \in \mathbb{F}_{q^k}^4$ and $ad \neq bc$. Write the factorization of f into monic irreducible polynomials as $f(Y) = \prod_{i=1}^{k} F_i(Y)^{e_i}$. It induces a factorization of F_{abcd}*

$$F_{abcd}(U) = \prod_{i=1}^{k} \left((cx+d)^{\deg F_i} F_i\left(\frac{ax+b}{cx+d}\right)\right)^{e_i}.$$

Note that the factors in this decomposition are not necessary monic, not necessary irreducible and may have a lower degree than the corresponding factor in F_i.

Proof. The induced factorization is clear. It suffices to perform the change of variable on both sides and remark that the grouped terms

$$(cx+d)^{\deg F_i} F_i\left(\frac{ax+b}{cx+d}\right)$$

are indeed polynomials.

It is also clear that the transformed factors have no reason to be irreducible in the extension field $\mathbb{F}_{q^k}$.

Remark that when $c \neq 0$ the coefficient of $x^{\deg F_i}$ in the factor coming from F_i is $c^{\deg F_i} F_i(a/c)$. Since this is not necessarily 1 and can even be 0, we see that the transformed polynomials are not necessarily monic and may have degree strictly smaller than the corresponding F_i. □

Thanks to this, it is now possible to amplify a single polynomial to a much larger extend than previously. More precisely, with a linear change of variable, number of amplified copies of a single polynomial is close to the size of finite field in which a is picked. With homographies, the number of copies becomes close to the cube of the size of the finite field.

Basic idea 2. The second idea directly stems from this fact. Since it is possible to make so many copies of one polynomial, it suffices to start from a single polynomial f. Thus, instead of considering many polynomials until we find some candidate, we are going to choose a polynomial with factors by design. Over a small finite field $\mathbb{F}_q$, an extremely natural candidate to consider is:

$$f(x) = x^q - x.$$

It is well-known that this polynomial splits into linear factors, since any element of $\mathbb{F}_q$ is a root of f.

The rest of the paper gives the details of how to put together these two basic ideas into a working discrete logarithm algorithm.

3 New algorithm: Setting

In this section, we introduce our new discrete logarithm algorithm for small characteristic fields. We first describe the general setting of our algorithm, before considering its relation collection phase. We skip the description of the linear algebra phase that takes as input the relations and outputs logarithms of the elements of our factor base(s), since it is left unchanged compared to previous algorithms. Finally, we study the computation of individual discrete logarithms, for arbitrary field elements. This phase relies on a descent strategy as in [5], which needs to be deeply adapted to the specificities of our new relation collection phase.

3.1 Choosing the parameters

The finite fields that we consider are of the form $\mathbb{F}_{q^{2k}}$, obtained by taking a degree k extension of a base field $\mathbb{F}_{q^2}$, which is not necessarily prime. In our construction, we only deal with cases where $k \leq q + \delta$, for some small offset δ (compared to q). The extension is constructed using a irreducible factor of a low degree bivariate polynomial, evaluated at (X, X^p). More precisely, we choose two low degree polynomials $h_0(X)$ and $h_1(X)$ with coefficients in $\mathbb{F}_{q^2}$ such that $h_1(X)X^q - h_0(X)$ has an irreducible factor $I(X)$ of degree k. Heuristically, we expect arbitrary extension degrees to appear with this form of definition polynomials. For example, we performed some experiments with $q = 101$ and found that all extension degrees up to 103 appeared with h_0 and h_1 of degree at most 2 and coefficients in $\mathbb{F}_{101}$. Indeed, we heuristically expect that a fraction close to $1/k$ of random polynomials has a factor of degree k. Thus considering polynomials h_0 and h_1 of degree 2, we have a very large number of degrees of freedom and expect to get a good representation. However, this argument is not a proof. In particular, with linear polynomials h_0 and h_1 we can only reach a fraction of the possible extension degrees (see the simple cases below).

Still, since we have the option of raising the degree of h_0 and h_1 to arbitrary constants, it should be easy to achieve any extension degree k (up to $q + \deg(h_1)$.)

Some simple cases. Some kind of extensions are especially well-suited to this form of representation. To illustrate our construction, we now describe these simple cases.

A first example concerns extensions of degree $k = q - 1$. They can be represented as Kummer extensions by an irreducible polynomial $I(X) = X^{q-1} + g$, where g is a generator of the multiplicative group $\mathbb{F}_p^*$. This can be achieved easily in our setting by letting $h_0(X) = -gX$ and $h_1(X) = 1$.

Similarly extensions of degree $k = q + 1$ can be represented by a Kummer extension by an irreducible polynomial $I(X) = X^{q+1} + G^{q-1}$, where G is a generator of the multiplicative group $\mathbb{F}_{q^2}^*$. This can be achieved easily in our setting by letting $h_0(X) = -G^{q-1}$ and $h_1(X) = X$.

Another special case is $k = p$, which can be represented by an Artin-Schreier extension with $I(X) = X^p - X - 1$. This can be achieved by choosing $h_0(X) = -(X + 1)$ and $h_1(X) = 1$.

Extending to smaller values of p. One apparent difficulty of our choice of parameters is that it only works when $k \leq p + \deg h_1$, i.e. k is not much bigger than p. In particular, this is problematic for binary field of the form $\mathbb{F}_{2^k}$ when the extension degree is prime.

To extend to this case, it suffices to start by embedding the base field $\mathbb{F}_p$ into $\mathbb{F}_q$ with $q = p^r$, the smallest power of p larger than k.

4 New algorithm: initial phase

In order to generate the relations, we start from the polynomial equation:

$$\prod_{\alpha \in \mathbb{F}_q} (Y - \alpha) = Y^q - Y, \tag{2}$$

and perform a change of variable $Y = \frac{aX+b}{cX+d}$, with $(a, b, c, d) \in \mathbb{F}_{q^2}^4$ satisying $ad - bc \neq 0$.

Evaluating Equation (2) and multiplying by $(cX + d)^{q+1}$, we find that:

$$(cX + d) \prod_{\alpha \in \mathbb{F}_q} ((a - \alpha c)X + (b - \alpha d)) = (cX + d)(aX + b)^q - (aX + b)(cX + d)^q.$$

Moreover the right-hand side can be evaluated to:

$$\begin{gathered}(ca^q - ac^q)X^{q+1} + (da^q - bc^q)X^q + (cb^q - ad^q)X + (db^q - bd^q) \equiv \\ \frac{(ca^q - ac^q)Xh_0(X) + (da^q - bc^q)h_0(X) + (cb^q - ad^q)Xh_1(X) + (db^q - bd^q)h_1(X)}{h_1(X)} \pmod{I(X)}.\end{gathered}$$

As a consequence, we get an equality in $\mathbb{F}_{q^{2^k}}$ between a product of linear polynomials and a fraction with low-degree numerator and constant denominator. Considering $h_1(X)$ as an extra element of the smoothness basis, we get a multiplicative relation whenever the numerator factors into linear factors.

Important note: The reader should be aware that the same relation may be encountered several times (with different quadruples (a, b, c, d)). In particular, when $(a, b, c, d) \in \mathbb{F}_q^4$, we obtain a trivial equation. This phenomenom is due to the structure of the homographies seen as matrices with determinant 1. It is the reason why we need to take coefficients[1] in $\mathbb{F}_{q^2}$ and to consider a smoothness basis with coefficients in $\mathbb{F}_{q^2}$.

[1] There is nothing really special with $\mathbb{F}_{q^2}$, it is just the smallest superfield of $\mathbb{F}_q$. A larger superfield would also work.

Cost of relation finding. With the structure of homographies in mind, we can choose the quadruple (a, b, c, d) in a way that avoids duplicate relations. We now would like to estimate the probability of finding a relation for a random quadruple. Under the usual heuristic that the right-hand side numerator factors into linear with a probability close to a random polynomial of the same degree, we need to perform $D!$ trials (where D denotes the degree of the right-hand numerator). We note that $D \leq 1 + \max(\deg h_0, \deg h_1)$.

As a consequence, we expect to find enough relations by considering $O(p^2)$ quadruples.

4.1 Extending the basis to degree 2 polynomials

The above process together with linear algebra can thus give us the logarithms of linear polynomials. Unfortunately, this is not enough. Indeed, at the present time, we do not know how to compute arbitrary logarihtms using only linear polynomials. Instead, we extend our basis of known logarithms to include polynomials of degree 2.

We propose two different strategies for completing the basis. The first strategy is the fastest, however, depending on the finite field and its description, it may not be able to access all polynomials. The second strategy is slower because it requires to do some additional linear algebra steps.

First strategy. The idea of this strategy is to reconstruct the relations produced for finding the linear polynomials. This time, instead of keeping the relations with a right-hand that splits into linear factors, we keep relations with a single degree 2 factor and some linear factors. This clearly allows us to compute the logarithm of the degree 2 polynomial.

Note that, depending on the degree of the right-hand side, it is possible to repeat this a few time. In each pass, we can learn some extra logarithms, as long as there are relations whose right-hand side factors logarithms are all known, expect a single degree 2 factor, which we then learn.

Second strategy. For this second strategy, we produce some extra equations, using the same approach with a different change of variable. More precisely, we consider changes of the form:

$$Y = \frac{aX^2 + bX + c}{dX^2 + eX + f}.$$

With this choice, the left-hand side factors into polynomials of degree at most 2. If the left-hand side also factors into polynomials of degree at most 2, we obtain an equation that involves the extended basis. Once we get enough equations, it suffices to perform a linear algebra step to recover the extra logarithms.

However, this linear algebra step is much bigger than the first one. Thankfully, it is often possible to improve this be considering a change a variable with a restricted form that lead to equations containing only a subset of the possible degree 2 polynomials.

Basically, the idea is to choose

$$Y = \frac{X^2 + aX + b}{X^2 + aX + c}.$$

With this choice, thanks to the repetition of $X^2 + aX$ in the numerator and denominator, all the degree 2 factors on the left are of the form $X^2 + aX + K$. If we only keep relations with a right-hand side that factors into linear polynomials, then, a set of relations that all share the same value for a produce a much smaller linear system. Indeed, the unknowns are the logarithms of irreducible polynomials of degree 2 of the restricted

form $X^2 + aX + K$, with a fixed a. As a consequence, instead of solving a large system of size $O(q^4)$, we need to solve q^2 smaller system (on for each a), of size $O(q^2)$.

Note that depending on the exact extension, it might be necessary to use different restricted form, with the same goal of splitting the system into smaller chunks in mind.

Finally, it is possible to reuse the first strategy on such reduced systems and complete our basis of degree 2 polynomials once we have obtained the logarithms values for one (or a few) subsets of degree 2 polynomials.

Depending on the exact parameters of the finite field and the number of logarithms that need to be computed, it might also be useful to further extend the smoothness basis and include polynomials of larger degree (3 or more).

5 New algorithm: descent phase

Once the logarithms of smoothness basis elements are known, we want to be able to compute the logarithm of an arbitrary element of the finite field. We wish to proceed using a descent approach similar to [5]. The basic idea is to first obtain a good representation of the desired element into a product of polynomials whose degrees are not too high. Then, proceeding recursively, we express the logarithms of those polynomials as sums of logarithms of polynomials of decreasing degree. Once we reach the polynomials of the smoothness basis, we are done.

In our context, we cannot, in general, use the preexisting method for this descent step. Yet, we first recall this method, discuss when it can be used and explain why we cannot use it generally. Then, we propose an alternative method that is more suited to the field representations we are using. This new method involves the resolution of bilinear multivariate systems of equations over $\mathbb{F}_{q^2}$. The resolution of such system has been analyzed careful in Spaenlehauer's PhD thesis [7].

5.1 Practical preliminary step

Before going into the descent itself, it is useful to start by finding a good representation of the element Z whose logarithm is desired. Initially, Z is expressed as a polynomial of degree up to $k-1$ over $\mathbb{F}_{q^2}$. Assuming that g denotes a generator of $\mathbb{F}_{q^{2k}}$, we consider the decomposition of the polynomials that represent $g^i Z$, until we find one which decomposes into elements of reasonably low degree. These lower degree elements are then processed by the descent step.

A classical improvement on this is to use a continued fraction algorithm to first express $g^i Z$ as a quotient of two polynomials of degree at most $k/2$.

This preliminary step gives no improvement on the asymptotic complexity of the descent phase.

5.2 Classical Descent method

The classical descent technique as described in [5] is based on Special-Q sieving. More precisely, it creates relations in a linear subspace where by construction one side of the equation is divisible by the desired polynomial.

It works in the context, recalled in Section 1.3, where we have two related variables X and Y. The relations are constructed by considering bivariate polynomials $h(X, Y)$, which can lead to relations of the form $h(X, f_1(X)) = h(f_2(Y), Y)$. To create a relation that involves a fixed polynomial $Q(X)$, we want to enforce the condition $h(X, f_1(X)) \equiv 0 \pmod{Q(X)}$. This condition is equivalent to $\deg(Q)$ linear equations on the coefficients of h. To get enough equations, it suffices to build the polynomials h as linear combination

of $\deg(Q)+2$ monomials. We, of course, choose monomials with low degree in X and Y and fix one arbitrary coefficient to 1.

In general characteristic, we cannot use this method in our context. However, with fixed characteristic fields, for example in characteristic 2 or 3, this become possible. Let p denote the charateristic of the finite field and let $Y = X^{p^r}$, where r is chosen to optimize the complexity of the descent. Then following our construction, we see that:

$$Y^{q \cdot p^{-r}} = \frac{h_0(X)}{h_1(X)}.$$

For the Kummer or Artin-Schreier cases, where h_0 and h_1 have degree at most one, this directly gives X as a polynomial in Y and the usual descent can be applied. When h_0 or h_1 have higher degree, the method still work, but we need to use a slight variation. Instead of considering the relation $h(X, f_1(X)) = h(f_2(Y), Y)$, we consider a relation $(h(X, f_1(X)))^{q \cdot p^{-r}} = h'(X^{q \cdot p^{-r}}, h_0(X)/h_1(X))$, where h' is obtained from h by raising the coeeficient to the power $q \cdot p^{-r}$. This has the additional advantage of completely eliminating the auxiliary variable Y.

In truth, this classical method is very efficient during the early However, by itself, this method cannot descend to very low degrees which is a problem when we want to keep a small smoothness basis. As a consequence, we combine it with a newer method described below, which works better on low degree polynomials.

5.3 New Descent method

The basic idea of the new descent method is given a polynomial Q of degree D, to find pairs of polynomials of degree $d = \lceil (D+1)/2 \rceil$, k_1 and k_2 such that $Q(X)$ divides $(k_1(x)^q k_2(x) - k_1(x)k_2(x)^q) \bmod I(X)$. As a consequence, the relation:

$$(k_1(x)^q k_2(x) - k_1(x)k_2(x)^q) \equiv (k_1(x)^q k_2(x) - k_1(x)k_2(x)^q) \bmod I(x),$$

has a factor equal to Q on the right-hand side and factors of degree at most d on the left-hand side. Since the total degree of the right-hand side is bounded by a small multiple of D (related to the degrees of h_0 and h_1 the polynomials which defined out extension field), with good probability, we obtain a relation between Q and polynomials of degree at most d.

The question is thus to construct such polynomials. We remark that the condition that $(h_1(x)^q h_2(x) - h_1(x)h_2(x)^q) \bmod I(X)$ vanishes modulo Q can be rewritten as a quadratic system of equations over $\mathbb{F}_q$. In fact, this system is even bilinear, since each monomial that appear in it contains at most one unknown for each of h_1 and h_2. As a consequence, this system can be quite efficiently solved using a Gröbner basis algorithm.

However, the cost of solving such system with currently existing algorithms remains exponential in d. As a consequence, for larger values of D, we need to use a different unbalanced split. More precisely, we take two polynomials of distinct degree d and $D+1-d$, to get enough degrees of freedom in the system. Since the cost of solving the resulting bilinear system of equations essentially depends on $\min(d, D+1-d) = d$ this allows us to perform descent of polynomials of arbitrary degree. It should be noted that we are now using a smaller rate of descent, which is going to greatly increase the number of nodes in the descent tree.

6 Complexity Analysis

6.1 Logarithms of the smoothness basis

We analyze the complexity of this phase under the heuristic assumption that our method can produce distinct and independent relations. We only consider the general case where

the full degree 2 smoothness basis, with coefficients in $\mathbb{F}_{q^2}$, is constructed as a whole. Forgetting about possible size reduction using the action of Frobenius, the size of the smoothness basis is $O(q^4)$. We thus need to construct $O(q^4)$ equations. The degree of the right-hand sides in our equation is bounded by $D_r = 1 + \max \deg(h_0), \deg(h_1)$. As a consequence, we need to consider $O(D_r! q^2)$ candidates in order to build relations. Solving the resulting linear systems of $O(q^4)$ equations of weight $O(q)$ in $O(q^4)$ has a complexity of $O(q^9)$ arithmetic operations modulo $q^{2k} - 1$ (or a divisor of this number).

The linear algebra part dominates if we can find a representation of our finite field that satisfies $D_r! \leq q^5$. Since, according to Section 3.1, we expect h_0 and h_1 to have constant degree, D_r is also constant and the condition $D_r! \leq q^5$ is clearly satisfied. As a consequence, the creation of the smoothness basis can be done in **polynomial time** in q and thus, this phase is polynomial in the size of the considered finite field.

A similar polynomial time behavior for computing the logarithms of the smoothness basis is also given in [1].

6.2 Individual logarithms

To analyze this phase, we use the simplifying hypothesis $q \approx k$. Under this hypothesis, remark that:

$$\begin{aligned} L_{q^{2k}}(\beta, c) &= \exp((c + o(1))(2k \log q)^{\beta} (\log(2k \log q))^{1-\beta}) \\ &\approx \exp((c' + o(1)) k^{\beta} \log(k)). \end{aligned}$$

We sketch the analysis of the complexity, ignoring the constant in the $L(\alpha, c)$. We show that we can reach complexity $L(1/4 + o(1))$ when the characteristic is fixed (such as 2 or 3). I.e., we have $q = p^{\ell}$ for some fixed value of p. Since $q \approx k$, we have ℓ, q and k growing to infinity.

We start with the classical descent approach, since it is compatible with our algorithm in fixed characteristic. The analysis of this method is well-known and we do not need to redo it completely. We simply remark that in order to descend to polynomials of degree $k^{1/2}$, it is possible to parametrize the classical descent in such a way that its cost is reduced to $L(1/4)$. This is not usually used, because stopping at polynomials of degree $k^{1/2}$ normally doesn't allow to finish the computation.

We then use the newer descent approach, starting from polynomials of degree $D \approx k^{1/2}$. To decompose each polynomial into polynomials of lower degree, we construct a relation using one polynomial of degree $d + 1$ and one of degree $D + 1 - d$. Since the degree of the right-hand side is a small multiple of D, a solution of the system yields with heuristic constant probability a new equation relating the desired polynomial to polynomials of degree at most $D - d$. The number of polynomials of degree between $D + 1 - 2d$ and $D - d$ is at most $q + O(1)$. Note that the contribution of lower degree polynomials is negligible, since the computation of their logarithms is deferred to a lower level of the computation tree, where they represent a tiny fraction of the polynomials to be addressed.

Thus, the running time to compute the logarithm of a degree D polynomial is $T(D, d) \approx T_0(D, d) + qT(D-d, d)$. According to [7], we have $T_0(D, d) = O(d\, D \binom{D+d+2}{d+2}^{\omega})$, where ω denotes the achievable exponent for matrix multiplication. We now choose d to ensure that $T_0(D, d)$ dominates the computation. This requires d to be large enough to be able to neglect the powers of q in the sum (when the expression of $T(D, d)$ is unrolled). To simplify the analysis, we replace T_0 by $T_1(D, d) = D^{3d+7}$, which is asymptotically larger. We find that we need to choose d such that $d(3d + 7) > D \log q$. Taking d above but close to this bound with $D \approx k^{1/2}$ yields $d = O(k^{1/4} \log^{1/2}(k))$. As a consequence, the cost of the newer part of the descent is bounded by $T_1(k^{1/2}, O(k^{1/4} \log^{1/2}(k)))$ per

polynomial. Due to the presence of an extra $\log^{1/2}(k)$ term, this is strictly bigger than $L(1/4)$. However, it can be rewritten as $L(1/4 + o(1))$.

Finally, since there are at most $L(1/4)$ polynomials of degree $k^{1/2}$ to consider, the overall complexity of the algorithm in fixed characteristic is $L(1/4 + o(1))$.

Impact of more efficient algorithms to solve the bilinear systems. It is important to remark that given an oracle (or efficient algorithm) to solve the bilinear systems, we could use a much faster descent from degree D to $\lceil (D+1)/2 \rceil$ at each step. In this case, the complexity would be dominated by the number of nodes in the descent tree, i.e. $q^{\log D}$. Starting directly from $D = k - 1$ would then give a **quasi-polynomial** complexity $\exp(\log^2 q)$.

6.3 Special case of $\mathbb{F}_{p^k}$, p and k prime

In order to use our algorithm, we need to embed $\mathbb{F}_{p^k}$ with p and k prime into $\mathbb{F}_{q^{2k}}$, with $q = p^\ell$ and $\ell = \lceil \log k \rceil$. From an asymptotic point of view, this is of little impact, indeed $L_{q^{2k}}(1/4 + o(1))$ can be rewritten as $L_{p^k}(1/4 + o(1))$ with a slightly larger $o(1)$.

7 Experiment on a Kummer extension $\mathbb{F}_{(2^{14})^{127}}$

We define $\mathbb{F}_{2^{14}}$ as $\mathbb{F}_2[a]$ with $a^14 + a^5 + 1 = 0$. Then, $\mathbb{F}_{2^{14 \cdot 127}}$ is obtained as a Kummer extension $\mathbb{F}_{2^{14}}[u]$ with $u^{127} = a + a^{128}$.

To compute our basis of discrete logarithms, we proceed in two parts as explained in Section 4. First we obtain the logarithm of the linear polynomials $u + \alpha$ with α in $\mathbb{F}_{2^{14}}$. Thanks to the action of Frobenius, this part of the basis can be reduced to 66 elements.

We then compute the logarithms of elements of the form $u^2 + \alpha u + \beta$. Using the action of Frobenius, we can multiply α by an arbitrary power of $A = a + a^{128}$. This allows us to split the degree 2 polynomials in 129 distinct subsets. Moreover, these subsets are further grouped into pairs which are again related by Frobenius (except the $u^2 + u + \beta$ which is related to itself and is twice as small). This allows us to recover the complete basis of logarithms by solving 64 systems on 8192 unknowns and one system on 4096 unknowns.

In [4], we announced that the linear algebra was done using 210 CPU.hours. Note however, that our linear algebra code is not really adapted to the tiny sizes of the considered system. As a consequence, this can probably be reduced.

The descent phase uses the mix of classical and newer descent that we described earlier. More precisely, starting with the preliminary phase, we transform the initial problem into a representation that contains polynomials of degree at most 18. We use the classical descent with $Y = X^8$ to lower the representation down to polynomials of degree at most 5. With the newer descent we can express polynomials of degree 4 and 5 using polynomials of degree 3 and polynomials of degree 3 using polynomials of degree 2, which concludes the computation. The total number of degree 3 polynomials that appeared during the computation is 86 803.

As mentionned in [4], the descent step took a little more than 5 CPU.hours.

Acknowledgements

We acknowledge that the results in this paper have been achieved using the PRACE Research Infrastructure resource Curie based in France at TGCC, Bruyères-le-Chatel (project number 2011050868) and the PRACE Research Infrastructure resource Jugene/Juqueen based in Germany at the Jülich Supercomputing Centre (project number 2011050868).

References

1. Faruk Gologlu, Robert Granger, Gary McGuire, and Jens Zumbragel. On the function field sieve and the impact of higher splitting probabilities: Application to discrete logarithms in $\mathbb{F}_{2^{1971}}$. Cryptology ePrint Archive, 2012. Report 2013/074.
2. Robert Granger, Faruk Gologlu, Gary McGuire, and Jens Zumbragel. Discrete logarithms in GF(2^{1971}). NMBRTHRY list, February 2013.
3. Antoine Joux. Faster index calculus for the medium prime case. Application to 1175-bit and 1425-bit finite fields. Cryptology ePrint Archive, 2012. Report 2012/720.
4. Antoine Joux. Discrete logarithms in GF(2^{1778}). NMBRTHRY list, February 2013.
5. Antoine Joux and Reynald Lercier. The function field sieve in the medium prime case. In Serge Vaudenay, editor, *EUROCRYPT'2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2006.
6. Daniel Panario, Xavier Gourdon, and Philippe Flajolet. An analytic approach to smooth polynomials over finite fields. In J. Buhler, editor, *Algorithmic Number Theory, Proceedings of the ANTS-III conference*, volume 1423, pages 226–236. Springer, 1998.
7. Pierre-Jean Spaenlehauer. *Solving multihomogeneous and determinantal systems Algorithms - Complexity - Applications*. PhD thesis, Université Pierre et Marie Curie (UPMC), 2012.