A New Index Calculus Algorithm with Complexity $L(1/4 + o(1))$ in Small Characteristic

Antoine Joux[✉]

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

Abstract. In this paper, we describe a new algorithm for discrete logarithms in small characteristic. This algorithm is based on index calculus and includes two new contributions. The first is a new method for generating multiplicative relations among elements of a small smoothness basis. The second is a new descent strategy that allows us to express the logarithm of an arbitrary finite field element in terms of the logarithm of elements from the smoothness basis. For a small characteristic finite field of size $Q = p^n$, this algorithm achieves heuristic complexity $L_Q(1/4 + o(1))$. For technical reasons, unless n is already a composite with factors of the right size, this is done by embedding $\mathbb{F}_Q$ in a small extension $\mathbb{F}_{Q^e}$ with $e \leq 2\lceil \log_p n \rceil$.

1 Introduction

The discrete logarithm problem is one of the major hard problems used in cryptography. In this paper, we show that for finite fields of small characteristic, this problem can be solved with heuristic complexity $L(1/4 + o(1))$. Moreover, the algorithm yields very practical improvements compared to the previous state-of-the-art.

One of the two main ideas used for our algorithm is a generalization of the pinpointing technique proposed in [10]. Another independent algorithm for characteristic 2 was proposed in [5], yielding an algorithm with complexity $L_Q(1/3)$, with a better constant than the Function Field Sieve.

2 A Reminder of Discrete Logarithm Algorithms in Small Characteristic

As usual when studying index calculus algorithms, we write:

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

where $Q = p^n$ denotes the size of the finite field.

T. Lange, K. Lauter, and P. Lisoněk (Eds.): SAC 2013, LNCS 8282, pp. 355–379, 2014.
DOI: 10.1007/978-3-662-43414-7_18, © Springer-Verlag Berlin Heidelberg 2014

When considering the computation of discrete logarithms, in fields of the form $\mathbb{F}_Q$, where p is relatively small compared to Q, the state of the art choice is to use one of the numerous variation of the function field sieve. For larger values of p, it becomes preferable to use a variation of the number field sieve. The choice between the two family of algorithms is made by comparing p and $L_Q(\frac{1}{3})$ (see [12]).

All these variants of the function field sieve find multiplicative relations by factoring various polynomials into polynomials of low degree. A 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 [13].

When using function field sieve algorithms, a standard *heuristic assumption* is to assume that all polynomials that arise in the algorithm also follow this smoothness probability. In the new algorithm presented here, 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: those for which there is no known reason to expect that they would deviate from the normal behavior.

Despite the new ingredients we are using, there are deep similarities between our algorithm and its predecessors. In particular, some features are reminiscent of Coppersmith's algorithm [3], while others are inspired from [11]. In the present section, we recall these two algorithms.

2.1 Coppersmith's Algorithm

Coppersmith's Algorithm was published in 1984 in [3]. Historically, it was the first discrete logarithm algorithm to achieve complexity $L(1/3)$. In its original presentation, this algorithm is dedicated to characteristic 2, but it can easily be generalized to any fixed characteristic [14].

Consider as usual a finite field of size $Q = p^n$. Coppersmith assumes that $\mathbb{F}_Q$ is constructed using a polynomial $P(x) = x^n - P_0(x)$, where $P_0(x)$ is a polynomial of low degree. He then chooses k a power of p close to $\sqrt{n}$ and writes $n = hk - n_0$, with $0 \leq n_0 < k$.

Let A and B be two polynomials of low degree. Coppersmith considers the polynomial $C(x) = x^h A(x) + B(x)$, let $D = C^k$ and remarks that since k is a power of p, the linearity of the Frobenius map implies:

$$\begin{aligned} D(x) &= C(x)^k \\ &= x^{hk} A(x)^k + B(x)^k \pmod{P(x)} \\ &= x^{n_0} P_0(x) A(x)^k + B(x)^k \pmod{P(x)} \end{aligned}$$

As a consequence, both C and D have moderate degrees $O(\sqrt{n})$. If both factors into low degree polynomials, we obtain a multiplicative relation between the factors; in this relation, the factors of C are raised to the power k.

The complexity of Coppersmith's index calculus algorithm is $L(1/3, c)$ with a value of c that depends on the extension degree n. This constant is minimized when n is close to a power of p^2.

2.2 Function Field Sieve

The general function field sieve algorithm was proposed in 1999 by Adleman and Huang in [1]. It improves on Coppermith's algorithm when the extension degree n is not close to a power of p^2. In its general form, it uses multiplicative relations between ideals in function fields and technicalities arise due to this. A simplified version was proposed in [11]. This simplified version only involves polynomial rings (instead of function fields), which has the merit of removing most of these technicalities.

The algorithm from [11] is particularly well suited to the computation of discrete logarithms in fields that contain a medium-sized subfield (not necessarily prime). To emphasize this, we write the finite field as $\mathbb{F}_{q^n}$, a degree n extension of the medium-sized field $\mathbb{F}_q$. In the optimal case where $q = L_{q^n}(1/3)$, the constant in the complexity can even be reduced compared to usual function field sieve.

By convention, in the rest of the section, X and Y are formal variables, while x and y are elements of $\mathbb{F}_{q^n}$. In order to define the extension field $\mathbb{F}_{q^n}$, the algorithm selects g_1 and g_2 two univariate polynomials of respective degree d_1 and d_2 with coefficients in $\mathbb{F}_q$. If the polynomial $-g_2(g_1(Y)) + Y$ has an irreducible factor $\mathcal{I}(Y)$ of degree n over $\mathbb{F}_q$, then $\mathcal{I}$ can be used to define $\mathbb{F}_{q^n}$ and we denote by y a root of $\mathcal{I}$ in this field. When this occurs, $-g_1(g_2(X)) + X$ also has an irreducible factor $\mathcal{I}'(X)$ of degree n over $\mathbb{F}_q$. Moreover, $x = g_1(y)$ is a root of $\mathcal{I}'$ in $\mathbb{F}_{q^n}$. Abstractly, we consider that the algorithm is, in fact, defining the finite field $\mathbb{F}_{q^n}$ implicitly by the two relations:

$$x = g_1(y), \qquad y = g_2(x), \tag{1}$$

As explained in [11], it is easy to find polynomials g_1 and g_2 that satisfy this requirement. This definition of the finite field induces the commutative diagram in Fig. 1. On the right-hand side, we use the $\mathcal{I}(Y)$ to define $\mathbb{F}_{q^n}$ and on the left-hand side, we use $\mathcal{I}'(X)$.

$$\begin{tikzcd}
& \mathbb{F}_q[X,Y] \arrow[dl, "Y\leftarrow g_2(X)"'] \arrow[dr, "X\leftarrow g_1(Y)"] & \\
\mathbb{F}_q[X] \arrow[dr, "X\leftarrow x"] & & \mathbb{F}_q[Y] \arrow[dl, "Y\leftarrow y"] \\
& \mathbb{F}_{q^n} &
\end{tikzcd}$$

Fig. 1. Commutative diagram for the algorithm of [11]

The relative degrees of d_1 and d_2 in the construction 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 $d_2 \approx \sqrt{n/D}$. The simplest and most efficient case occurs when we can choose $D = 1$, i.e. $d_1 \approx d_2 \approx \sqrt{n}$.

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 two univariate polynomials whose images in $\mathbb{F}_{q^n}$ are equal, i.e. an equation:

$$\mathcal{A}(y)\,g_1(y) + \mathcal{B}(y) = \mathcal{A}(g_2(x))\,x + \mathcal{B}(g_2(x)).$$

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

Following [11] , we only keep the relations, where $\mathcal{A}(Y)\,g_1(Y) + \mathcal{B}(Y)$ and $\mathcal{A}(g_2(X))\,X + \mathcal{B}(g_2(X))$ both factor into unitary polynomials of degree at most D in X or Y. This yields multiplicative relations between the images in $\mathbb{F}_{q^n}$ of these low-degree polynomials, which form the smoothness basis. Classically the good pairs $(\mathcal{A}, \mathcal{B})$ are found using a sieving approach[1].

Complexity. To express the complexity, [11] let $Q = q^n$ and assumes that there exists a parameter α such that:

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

In this setting, the heuristic asymptotic complexity of the sieving phase is $L_{q^n}(\frac{1}{3}, c_1)$ and the complexity of the linear algebra is $L_{q^n}(\frac{1}{3}, c_2)$, 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 that we can obtain enough linear equations to build the linear system of equations. This requires:

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

For a given finite field $\mathbb{F}_{q^n}$, [11] indicates that the best possible complexity is obtained by choosing the smallest acceptable value for the parameter D.

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

[1] Asymptotically, exhaustive search of good pairs is as efficient. However, using sieving improves things by a large constant factor.

We now detail this phase in the case of [11]. The ultimate goal of expressing a given element as a product of elements from the smoothness basis is not achieved in a single pass. Instead, it is done by first expressing the desired element in $\mathbb{F}_{q^n}$ as a product of univariate polynomials in either x or y and with degree smaller than that of the desired element. These polynomials can in turn be related to polynomials of a lower degree and so on, until hitting degree $\leq D$, i.e. elements of the smoothness basis. For this reason, the individual logarithm phase is also called the descent phase.

In order to create relations between a polynomial in either X or Y (i.e. coming either from the left or right side of a previous equation) and polynomials of lower degree, [11] proceeds as follows: Let $\mathcal{Q}(X)$ (resp. $\mathcal{Q}(Y)$) denote the polynomial that represent the desired element. One considers a set of monomials $S_{\mathcal{Q}} = \{X^iY^j | i \in [0 \cdots D_x(\mathcal{Q})], j \in [0 \cdots D_y(\mathcal{Q})]\}$, where $D_x(\mathcal{Q})$ and $D_y(\mathcal{Q})$ are parameters that we determine later on. Each monomial in S_Q can be expressed as a univariate polynomial in X (resp. Y), after replacing Y by $g_2(X)$ (or X by $g_1(Y)$). For a monomial m we denote by $V_{\mathcal{Q}}(m)$ the value modulo $\mathcal{Q}$ of the univariate polynomial corresponding to m. Clearly, $V_{\mathcal{Q}}(m)$ can be represented by a vector of $\deg \mathcal{Q}$ finite field elements. We now build a matrix $M_{\mathcal{Q}}$ by assembling all the vectors $V_{\mathcal{Q}}(m)$ for $m \in S_{\mathcal{Q}}$. Any vector in the kernel of $M_{\mathcal{Q}}$ can then be interpreted as a polynomial whose univariate representation is divisible by $\mathcal{Q}$. If both the quotient after dividing by $\mathcal{Q}$ and the univariate representation in the other unknown decompose into products of polynomials of low enough degree, we obtain the desired relation.

Clearly, this approach requires us to take enough monomials to make sure that the kernel contains sufficiently many polynomials in order to find a satisfying relation. This can be achieved by choosing $D_x(\mathcal{Q})$ and $D_y(\mathcal{Q})$ such that $D_x(\mathcal{Q})D_y(\mathcal{Q}) \geq \deg \mathcal{Q}$. Moreover to balance the degrees after replacing X or Y, we make sure that $D_y(\mathcal{Q})/D_x(\mathcal{Q}) \approx D$. With these choices, the degree on each side after replacement is close to $\sqrt{n \deg \mathcal{Q}}$. The logarithm of the probability that each of the two sides decompose into polynomials of degree at most $\mu \deg \mathcal{Q}$ (after factoring out $\mathcal{Q}$) is estimated by:

$$-\frac{2}{\mu}\sqrt{\frac{n}{\deg \mathcal{Q}}}\log\left(\frac{2}{\mu}\sqrt{\frac{n}{\deg \mathcal{Q}}}\right).$$

The cost of this descent step increases when the degree of $\mathcal{Q}$ decreases. As a consequence, the total cost of the descent is dominated by the lowest degree polynomials that still need to be processed. In [11], the descent is used all the way down to constant degree D. As a consequence, with the relative sizes of n and q that [11] considers, the asymptotic complexity of the descent is $L_Q(1/3)$.

3 New Algorithm: Basic Ideas

The new index calculus algorithms proposed in this paper hinges on a few basic ideas, which can be arranged into a functional discrete logarithm algorithm.

Basic idea 1: Homographies. In [10], it was remarked that a single 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 non-zero 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 induced by 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 $F_{abcd}(X) = (cX+d)^{\deg f} f\left(\frac{aX+b}{cX+d}\right)$ 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}*

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

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. $\square$

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 variables, the number of amplified copies of a single polynomial is close to the size of the finite field in which a is picked. With homographies, the number of copies becomes larger (see Sect. 4.2 for a detailed analysis).

Basic idea 2: Systematic polynomial splitting. 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.

Geometrically, using the homogeneous evaluation of f (with multiplication by $(cX + d)^{q+1}$) at an homography h is equivalent to considering the image of the projective line (including its point at infinity) $\mathbb{P}_1(\mathbb{F}_q)$ by h.

Basic idea 3: Field definition The image of $X^q - X$ by an homography is a polynomial which only contains the monomials X^{q+1}, X^q, X and 1. To obtain a multiplicative relation, it is thus desirable to find a finite field representation that transforms such a polynomial into a low degree polynomial. This can be achieved by choosing the finite field representation in a way that is reminiscent both of Coppersmith's algorithm and of [11].

More precisely, we ask for a relation in $\mathbb{F}_{q^n}$ of the form:

$$x^q = \frac{h_0(x)}{h_1(x)}.$$

This defines the finite field $\mathbb{F}_{q^n}$, if and only if, $h_1(X)X^q - h_0(X)$ admits an irreducible factor of degree n.

This construction is similar to Coppersmith's Algorithm, since we require a simple expression of the Frobenius map $x \to x^q$. It is similar to [11], because we do not ask for the relation to directly give an irreducible polynomial but only require a factor of the proper degree.

The rest of the paper gives the details of how to put together these basic ideas into a working discrete logarithm algorithm. Another application of the same ideas has been described in [7], where a new deterministic algorithm (based on similar heuristics to ours) is proposed to find a provable multiplicative generator of a finite field.

4 Description of the New Algorithm

In this section, we present the new discrete logarithm algorithm for small characteristic fields that arises when putting together the basic ideas from the previous section. 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 basis, 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 method which contains two main strategies. For elements with representations of high degree, one proceeds as in [11], while for lower degrees we introduce a new strategy based on the resolution of multivariate systems of bilinear equations.

4.1 Choosing the Parameters

Given a small characteristic finite field $\mathbb{F}_{p^n}$, we start be embedding it into a field of the form $\mathbb{F}_{q^{2k}}$, with $k \leq q$. This can be achieved by taking a degree e extension of $\mathbb{F}_{p^n}$, with $e \leq 2\lceil \log_p n \rceil$.

After this initial embedding, the finite field $\mathbb{F}_{q^{2k}}$ is constructed has a degree k extension of $\mathbb{F}_{q^2}$. This degree k extension is obtained by using a irreducible factor of a low degree bivariate polynomial, evaluated at (X, X^p). More precisely, we follow the third idea of Sect. 3 and 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 $\mathcal{I}(X)$ of degree k. This field representation leads to the commutative diagram in Fig. 2. *Heuristically,* we expect arbitrary extension degrees to appear with this form of definition polynomials. Indeed, we 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. This is emphasized by the fact that with linear polynomials h_0 and h_1 we can only reach a fraction of the possible extension degrees (see the simple cases below).

In order to increase the confidence level in this heuristic hypothesis, we have performed some experiments in characteristic 2. This yields some numerical evidence supporting the heuristic which is described in Appendix B.

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

$$\begin{tikzcd}
& \mathbb{F}_q[X, X'] \arrow[dl, "X' \leftarrow X^q"'] \arrow[dr, "X' \leftarrow h_0(X)/h_1(X)"] & \\
\mathbb{F}_q[X] \arrow[dr, "X \leftarrow x"] & & \mathbb{F}_q[X] \arrow[dl, "X \leftarrow x"'] \\
& \mathbb{F}_{q^{2k}} \arrow[d, dash, "\text{Degree } e \text{ extension}"] & \\
& \mathbb{F}_{p^n} &
\end{tikzcd}$$

Fig. 2. Commutative diagram for our new algorithm

Illustration of 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 $\mathcal{I}(X) = X^{q-1} - g$, where g is a generator of the multiplicative group $\mathbb{F}_q^*$. 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 $\mathcal{I}(X) = X^{q+1} + G^{q-1}$, 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 $h_1(X) = X$.

Alternatively, extensions of degree $q + 1$ can also be represented using a "twisted Kummer" form, i.e. using an irreducible polynomial of the form $X^{q+1} - AX - B$, with coefficients A and B in $\mathbb{F}_q$. This twisted Kummer form is used for the record computation presented in Sect. 8.

Another special case is $k = p$, which can be represented by an Artin-Schreier extension with $\mathcal{I}(X) = X^p - X - 1$. This can be achieved by choosing $h_0(X) = -(X + 1)$ and $h_1(X) = 1$. However, since our algorithm is dedicated to small characteristic, this only leads to low degree extensions.

Action of Frobenius. When using Kummer, twisted Kummer or Artin-Schreier extensions, the fact that the Frobenius maps X to an homography in X allows to reduce the size of the factor basis by a factor close to the extension degree. Indeed, we can remark that:

1. In the Kummer case:

$$(x + \theta)^q = x^q + \theta^q = g(x + \theta^q/g).$$

2. In the twisted Kummer case:

$$(x + \theta)^q = x^q + \theta^q = \frac{1}{x}((\theta + A)x + B).$$

3. In the Artin-Schreier case:

$$(x + \theta)^p = x^p + \theta^p + 1.$$

These relations yield simple linear relations between the elements of the smoothness basis and allow to reduce its size. It is easy to check that similar relations also relate polynomials of degree 2 in x.

4.2 Logarithms of Linear Polynomials

The first step in the algorithm is to generate the logarithms of all linear polynomials in the finite field. As usual in index calculus, we construct multiplicative relations between these elements. These equations can be viewed as linear equations on the values of their logarithms which are then obtained by linear algebra.

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

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

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

Evaluating Eq. (3) 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:

$$\begin{gathered}(ca^q - ac^q)X^{q+1} + (da^q - bc^q)X^q + (cb^q - ad^q)X + (db^q - bd^q) \\ \equiv \\ \frac{(ca^q - ac^q)Xh_0(X) + (da^q - bc^q)h_0(X) + (cb^q - ad^q)Xh_1(X) + (db^q - bd^q)h_1(X)}{h_1(X)} \\ \pmod{\mathcal{I}(X)}.\end{gathered} \tag{4}$$

As a consequence, we get an equality in $\mathbb{F}_{q^{2k}}$ 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 right-hand side's numerator factors into linear factors.

Counting the Relation Candidates. The above description that generates a candidate relation from a quadruple (a, b, c, d) disregards some important structure in the set of candidates. In particular, the reader should be aware that the same relation may be encountered several times (with different quadruples (a, b, c, d)). Typically, when $(a, b, c, d) \in \mathbb{F}_q^4$, we obtain a trivial equation. Indeed, in this case, for each v in $\{a, b, c, d\}$, we have $v^q = v$. As a consequence, after evaluation, we obtain the polynomial $(ad - bc)(X^q - X)$. Since this a just a multiple of the basic polynomial $X^q - X$, this cannot form a new relation.

This is the reason why we need to take coefficients[2] (a, b, c, d) in $\mathbb{F}_{q^2}$ and to consider a smoothness basis with coefficients in $\mathbb{F}_{q^2}$.

To understand the way Eq. (3) is transformed, we need to recall a few facts about the geometric action of homographies on a projective line. Given a nonsingular matrix M with coefficients in a field $\mathbb{K}$ and a point P on the projective line $\mathbb{P}_1(\mathbb{K})$ with homogeneous coordinates (X_P, Y_P), we define the image of P by M to be the point MP, whose coordinates are obtained by multiplying the matrix M and the vector of coordinates of P. In other words, when

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

[2] At this point, 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.

MP is defined as the point with homogeneous coordinates $(aX_P + bY_P, cX_P + dY_P)$. It is clear that multiplying M by an arbitrary non-zero scalar does not change the induced geometric action. Since the scalar matrices form a normal subgroup of the invertible matrices, it is better to consider M as an element from the corresponding quotient group which is traditionally called $\mathrm{PGL}_2(\mathbb{K})$. It is well known that for a finite field $\mathbb{K}$, the cardinality of $\mathrm{PGL}_2(\mathbb{K})$ is $(|\mathbb{K}|^2-1)\cdot|\mathbb{K}|$.

Geometrically, the change of variables we considered in the Eq. (3) is equivalent to writing a polynomial equation for the image of the projective line $\mathbb{P}_1(\mathbb{F}_q)$ by an element of $\mathrm{PGL}_2(\mathbb{F}_{q^2})$. Since $\mathrm{PGL}_2(\mathbb{F}_q)$ leaves $\mathbb{P}_1(\mathbb{F}_q)$ globally invariant, it is now clear that the same equation can arise multiple times. More precisely, for any M in $\mathrm{PGL}_2(\mathbb{F}_{q^2})$ and any N in $\mathrm{PGL}_2(\mathbb{F}_q)$, M and NM send $\mathbb{P}_1(\mathbb{F}_q)$ to the same image and induce the same equation. Since $\mathrm{PGL}_2(\mathbb{F}_q)$ is not a distinguished subgroup of $\mathrm{PGL}_2(\mathbb{F}_{q^2})$, we cannot form a quotient group. Instead, we need to regroup the matrices of $\mathrm{PGL}_2(\mathbb{F}_{q^2})$ into orbits induced by the (left-)action of $\mathrm{PGL}_2(\mathbb{F}_q)$. One can check that this action is free and thus that the cardinality of the set of orbits is simply the quotient of the cardinalities of $\mathrm{PGL}_2(\mathbb{F}_{q^2})$ and $\mathrm{PGL}_2(\mathbb{F}_q)$. As a consequence, the number of orbits and, thus, of candidate equations is:

$$\frac{q^6 - q^2}{q^3 - q} = q^3 + q.$$

Cost of relation finding. 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's 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, since D can heuristically be chosen as a constant, we expect to find enough relations by considering $O(q^2)$ quadruples. To avoid duplicates, we can either use the structure of the orbits of $\mathrm{PGL}_2(\mathbb{F}_{q^2})$ under the action of $\mathrm{PGL}_2(\mathbb{F}_q)$ or simply keep hash values of the relations to remove collisions. Since we only need $O(q^2)$ distinct elements in a set of size close to q^3, the number of expected collisions between relations for randomly chosen quadruples (a, b, c, d) is $O(q)$. As a consequence, dealing with these collisions does not induce any noticeable slow-down.

4.3 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, we do not know, in general, how to compute arbitrary logarithms using a smoothness basis that only contains linear polynomials. Instead, we extend our basis of known logarithms to include polynomials of degree 2.

We first describe a natural approach that does not work, before proposing a successful approach that requires some additional linear algebra steps.

A Natural Strategy that Fails. The idea of this strategy is to reconsider the relations produced for finding the linear polynomials. But, instead of keeping the relations with a right-hand side that splits into linear factors, it also keeps relations with a single degree 2 factor and some linear factors. This clearly allows us to compute the logarithm of the degree 2 polynomial.

A simple counting argument shows that in general, this approach must fail. Indeed, on the one-hand, the number of quadratic polynomials with coefficients in $\mathbb{F}_{q^2}$ is $O(q^4)$, while on the other hand, the number of relations that can be obtained from homographies with coefficients in $\mathbb{F}_{q^2}$ is close to q^3 when removing the duplicates arising from homographies with coefficients in $\mathbb{F}_q$. As a consequence, it is not possible to derive the logarithms of all quadratic polynomials in this way.

It is interesting to note that if we replace the field $\mathbb{F}_{q^2}$ by a larger field for the coefficients of the homographies, the natural approach becomes workable. However, we can see that the same counting argument shows that, in general, using $\mathbb{F}_{q^3}$ is not good enough since a fraction of the equation are lost due to the right-hand side splitting probability. Thus, to be able to recover the degree 2 polynomials with the simple strategy we need to, at least, use $\mathbb{F}_{q^4}$ as our basefield. Since the number of linear polynomials in this case is q^4, it is clearly preferable to stay with $\mathbb{F}_{q^2}$ and pay the price of constructing quadratic polynomials with the strategy below.

A Working Strategy. For this second strategy, we produce some extra equations, using the same approach as for linear polynomials together with a slightly more general 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. In fact, it is almost as expensive as initially building all linear polynomials over the larger basefield $\mathbb{F}_{q^4}$. Thankfully, it is often possible to improve this, by separating the degree 2 polynomials into several subsets which can be addressed independently.

Basically, the idea is to choose

$$Y = \frac{a(X^2 + \alpha X) + b}{c(X^2 + \alpha X) + d}.$$

With this choice, thanks to the repetition of $X^2 + \alpha X$ in the numerator and denominator, all the degree 2 factors on the left are of the form $X^2 + \alpha X + K$. If we only keep relations with a right-hand side that factors into linear polynomials, a set of relations that all share the same value for α then produce a much smaller linear system. Indeed, the unknowns are the logarithms of irreducible polynomials of degree 2 from the subset $X^2 + \alpha X + K$, with a fixed α. 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 α), of size $O(q^2)$. This system is obtained by selecting equations with a smooth left-hand side in a set of $O(q^3)$ candidates.

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 higher 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 [11]. 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 systems has been analyzed carefully in Spaenlehauer's PhD thesis [15] and in [4].

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 [11] and recalled in Sect. 2.2 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.

In the description of this method, 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 $\mathcal{Q}(X)$, we want to enforce the condition $h(X, f_1(X)) \equiv 0 \pmod{\mathcal{Q}(X)}$. This condition is equivalent to $\deg(\mathcal{Q})$ linear equations on the coefficients of h. When the basefield is not too small, to get enough equations, it suffices to build the polynomials h as linear combination of $\deg(\mathcal{Q}) + 2$ monomials.

In general characteristic, we cannot use this method in our context, because we do not known how to create two related variables X and Y to use in this descent step. However, with small characteristic fields, this become possible. Let p denote the characteristic of the finite field. We can then write $q = p^\ell$ and let $Y = X^{p^r}$, where $r = \lfloor \ell/2 \rfloor$. Then following our construction, we see that:

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

For the Kummer (or Artin-Schreier) case, where h_0 and h_1 have degree at most one, this directly gives X as a polynomial g in Y and the usual descent can be applied without modification. When h_0 or h_1 have higher degree, the method still works, but we need to use a slight variation. Instead of considering the relation $h(X, X^{p^r}) = h(g(Y), Y)$, we consider a relation $(h(X, X^{p^r}))^{q \cdot p^{-r}} = h'(X^{q \cdot p^{-r}}, h_0(X)/h_1(X))$, where h' is obtained from h by raising the coefficient to the power $q \cdot p^{-r}$. This has the additional advantage of completely eliminating the auxiliary variable Y.

As seen in Sect. 2.2, this becomes less and less efficient as the degree of $\mathcal{Q}$ decreases and the complexity is dominated by the lowest degree of $\mathcal{Q}$ that we consider.

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 Bilinear System Based Descent

The basic idea of the new descent method we propose to complement the classical descent works as follows: given a polynomial $\mathcal{Q}$, we search for a pair of polynomials of lower degree, k_1 and k_2 such that $\mathcal{Q}(X)$ divides $(k_1(X)^q k_2(X) - k_1(X) k_2(X)^q) \bmod \mathcal{I}(X)$. 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 $\mathcal{Q}$ on the right-hand side and factors of degree at most $D_m = \max(\deg k_1, \deg k_2)$ on the left-hand side. Since the total degree of the right-hand side is bounded by a small multiple of D_m (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 $\mathcal{Q}$ and polynomials of degree at most D_m.

The question is thus to construct such polynomials k_1 and k_2. We remark that the condition that $(k_1(X)^q k_2(X) - k_1(X)k_2(X)^q) \bmod \mathcal{I}(X)$ vanishes modulo $\mathcal{Q}$ can be rewritten as a quadratic system of multivariate 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 k_1 and k_2. As a consequence, this system can be quite efficiently solved using a Gröbner basis algorithm. More precisely, consider each coefficient of k_1 and k_2 as a formal unknown belonging to the field of coefficients $\mathbb{F}_{q^2}$. If x is one of these unknowns, we express x as $x_0 + zx_1$, where $(1, z)$ is a polynomial basis for $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$, x_0 and x_1 are unknowns belonging to $\mathbb{F}_q$. With this convention, we have $x^q = x_0 + z^q x_1$ and we can check that our polynomial system of equations is indeed bilinear over $\mathbb{F}_q$. This system contains $\deg \mathcal{Q}$ equations over $\mathbb{F}_{q^2}$ which are rewritten as $2 \deg \mathcal{Q}$ equations over $\mathbb{F}_q$. Assuming k_1 to be unitary, the maximal number of unknowns that can fit in k_1 and k_2 is $2(\deg k_1 + \deg k_2 + 1)$. However, due to the action of $\mathrm{PGL}_2(\mathbb{F}_q)$, several distinct pairs k_1, k_2 yield the same polynomial for $(k_1(X)^q k_2(X) - k_1(X)k_2(X)^q)$. To avoid this issue, we need to fix at least one of the unknowns over $\mathbb{F}_{q^2}$ to an element of $\mathbb{F}_{q^2} - \mathbb{F}_q$. After this, the number of remaining unknowns over $\mathbb{F}_q$ is $2(\deg k_1 + \deg k_2)$.

At this point, we need a new heuristic argument concerning the resulting system of equations. Namely, we require two important properties of the system that arise after fixing any additional unknowns to values. The resulting system is bilinear and its number of unknowns N is equal to its number of equations. We ask that with good probability this system should be zero-dimensional with at least one solution with values in the finite field $\mathbb{F}_q$. In order to apply this heuristic, we need at least one extra unknown over $\mathbb{F}_{q^2}$ that can be set to a random value. As a consequence, we require $\deg k_1 + \deg k_2 \geq \deg \mathcal{Q} + 1$.

Under this heuristic, we can analyze the cost of the bilinear descent by studying the complexity of solving one such system. The main result from [4,15] is that this complexity is exponential in $\min(\deg k_1, \deg k_2)$. For this reason, we do not use our descent strategy with balanced degrees $\deg k_1 \approx \deg k_2$), instead we let $d = \deg k_2$ parametrize the smallest of the two degrees and fix $\deg k_1 = \deg \mathcal{Q} + 1 - d$.

We recall the complexity analysis given in [4,15]:

Theorem 2 [*Corollary 3 from [4]*]**.**
The arithmetic complexity of computing a Gröbner basis of a generic bilinear system $f_1, \cdots, f_{n_x+n_y} \in \mathbb{K}[x_0, \cdots, x_{n_x-1}, y_0, \cdots, y_{n_y-1}]$ with the F_5 algorithm is upper bounded by:

$$O\left(\binom{n_x + n_y + \min(n_x + 1, n_y + 1)}{\min(n_x + 1, n_y + 1)}^{\omega}\right),$$

where $2 \leq \omega \leq 3$ is the linear algebra constant.

In our application, we have $n_x = 2(\deg \mathcal{Q}+1-d)$, $n_y = 2d$ and $\min(n_x, n_y) = 2d$. Thus, the cost of one descent step becomes:

$$\binom{2\deg \mathcal{Q}+3}{2d+1}^{\omega}.$$

An asymptotic choice for d is given in Sect. 6. It is obtained by making d large enough to make sure that the top level nodes of the descent tree dominate the total cost. Note that, in practical computations, the best choice is usually to make d as large as feasible. Indeed, the feasibility of the Gröbner step mostly depends on the available amount of memory and it is important to descent as steeply as possible to minimize the total cost.

6 Complexity Analysis

According to the heuristic argument of Sect. 4.1, the creation of the finite field representation runs in randomized polynomial time, just by trying random polynomials h_0 and h_1 of degree 2 (or higher constant degree). Similarly, the creation of the logarithms of linear and quadratic elements can be done in polynomial time. The dominating part of this initial creation of logarithms is dominated by the linear algebra required for the quadratic elements. Since we are solving q^2 linear systems of dimension $O(q^2)$ with $O(q)$ entries per line, the total cost of this polynomial part is $O(q^7)$ arithmetic operations. Note that for Kummer extensions, the number of linear systems is reduced to $O(q)$, which lowers the cost to $O(q^6)$.

A similar polynomial time behavior for computing the logarithms of the smoothness basis is also given in [5].

The rest of this section analyzes the descent phases which dominates the asymptotic cost of our algorithm.

6.1 Individual Logarithms

To analyze this phase it is convenient to write $k = \alpha q$, for some constant $\alpha \leq 1 + \frac{\deg h_1}{q}$. Under this hypothesis, remark that:

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

We now give the analysis of the complexity, which shows that we can reach complexity $L(1/4 + o(1))$ when the characteristic is small enough. Namely, we require $q = p^{\ell}$ for some $\ell \geq 2$.

We start with the classical descent approach, which it is compatible with our algorithm when $\ell \geq 2$. The analysis of this method is recalled in Sect. 2.2. Since the cost increases when $\deg \mathcal{Q}$ decreases, it suffices to write the cost for the lowest degree we wish to attain, namely $c_c\sqrt{q/\log q}$ for some constant c_c. The total cost in this case becomes:

$$\exp\left(\frac{1}{2\mu}\sqrt{\frac{\alpha}{c_c}}q^{1/4}\log q^{5/4}\right),$$

where $\mu < 1$.

Of course stopping at polynomials of degree $O(q^{1/2})$ is not enough to finish the computation. To continue the descent, we use the newer approach, starting from polynomials of degree $\deg \mathcal{Q} = c_c\sqrt{q/\log q}$. We need to determine the value of the parameter $d = \deg k_2$ introduced in Sect. 5.3. The left-hand side in the bilinear descent contains $q+1$ polynomials of degree at most $\deg k_1 = \deg \mathcal{Q} + 1 - d$. The degree of the right-hand side is bounded by $\deg k_1(\max(\deg h_0, \deg h_1) + 1)$, i.e., by a small multiple of $\deg \mathcal{Q}$, a solution of the system yields with heuristic constant probability a new equation relating the desired polynomial to polynomials of degree at most $\deg k_1$. The total number of polynomials of degree between $\deg k_1 - d$ and $\deg k_1$ after decomposing each side into irreducible polynomial is at most $q + O(1)$. Note that the contribution of lower degree polynomials to the complexity 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 $D_Q = \deg \mathcal{Q}$ polynomial is $T(D_Q, d) \approx T_0(D_Q, d) + qT(D_Q - d, d)$. In Sect. 5.3, we find that:

$$T_0(D_Q, d) = \binom{2D_Q + 3}{2d + 1}^{\omega}.$$

We now choose d to ensure that $T_0(D_Q, d)$ 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 $T(D_Q, d)$ is unrolled). To simplify the analysis, we replace T_0 by $T_1(D_Q, d) = D_Q^{6(d+1)}$, which is asymptotically larger. We find that we need to choose d such that:

$$q\,(D_q - d)^{6(d+1)} \leq {D_Q}^{6(d+1)}.$$

Taking the logarithm, and using $-\log(1-\epsilon) \approx \epsilon$, it asymptotically suffices to have

$$d^2 \geq \frac{D_Q \log q}{6}.$$

With $D_Q = c_c\sqrt{q/\log q}$, we can choose $d = \left\lceil \left(\frac{c_c}{6}\sqrt{q\log q}\right)^{1/2}\right\rceil$.

This yields:

$$T_1(D_Q, d) = \exp\left(\left(\frac{\sqrt{6c_c}}{4} + o(1)\right) q^{1/4}\log^{5/4} q\right).$$

Of course, this cost should be multiplied by the number of polynomials after the classical descent. When $\mu < 1$, the number of levels in the classical descent tree is logarithmic in q and each level multiplies the number of polynomials by a constant. As a consequence, the total number of polynomials after the classical descent is polynomial in q and vanishes into the $o(1)$ in the exponent of the complexity. In order the balance the two phases of the descent, we can take:

$$c_c = \frac{1}{\mu}\sqrt{\frac{2\alpha}{3}},$$

which achieves complexity:

$$\exp\left(\left(\frac{1}{2\sqrt{\mu}} \cdot \left(\frac{3\alpha}{2}\right)^{1/4} + o(1)\right) q^{1/4}\log^{5/4} q\right).$$

The constant in the above complexity could be improved by taking into account a value of the linear algebra constant $\omega < 3$ and letting μ tend toward 1. Note that due to the presence of an extra $\log^{1/2}(q)$ term, this is strictly bigger than $L(1/4)$. However, it can be rewritten as $L(1/4 + o(1))$.

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 $\deg \mathcal{Q}$ to $\lceil(\deg \mathcal{Q} + 1)/2\rceil$ 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 $\deg \mathcal{Q} = k - 1$ would then give a **quasi-polynomial** complexity $\exp(O(\log^2 q))$.

Moreover, this would get rid of the use of classical descent, together with the constraint of having $q = p^\ell$, with $\ell \geq 2$.

7 Remarks on the Special Case of $\mathbb{F}_{p^k}$, p and k Prime

As already mentioned, in order to use our algorithm, we need to embed $\mathbb{F}_{p^k}$ with p and k prime into a small extension $\mathbb{F}_{q^{2k}}$, with $q = p^e$ and $e = 2\lceil \log k \rceil$. From an asymptotic point of view, this is of little impact, indeed the complexity would become:

$$\exp\left(Ck^{1/4}\log^{5/4} k\right),$$

for some constant C. Since $\log p^k = k \log p \geq k/2$, expressed as a function of p^k, it becomes:

$$\exp\left(C' \log^{1/4} p^k \log\log^{5/4} p^k\right) = L_{p^k}(1/4 + o(1)).$$

In practice, it is also interesting to consider computations in $\mathbb{F}_{2^p}$ with $1024 < p < 2048$ prime. We know from Appendix B that this can be done by taking $q = 2^{11}$ and having polynomials h_0 and h_1 of degree 2. In this case, we expect the complexity to be dominated by the computation of logarithms of quadratic polynomials. This would require approximately 2^{77} arithmetic operations on numbers of p bits, since we only need the value of logarithms modulo $2^p - 1$. Comparing with the most recent data of the function field sieve [2], this $L(1/3)$ algorithm remains more efficient in this range.

8 A Couple of Experiments on Kummer Extensions in Characteristic 2

For practical experiments, it is very convenient to use finite fields containing a subfield of adequate size and to chose an extension that can be represented with polynomials h_0 and h_1 of degree 1. In practice, this means choosing a Kummer or twisted Kummer extension, which also a reduction of the size of the smoothness basis by a nice factor. We recently announce two computation records that illustrate the algorithm described here in this context. For numerical details about these records, we refer the reader to [8,9].

8.1 A Kummer Extension $\mathbb{F}_{256^{2\cdot 255}}$

Our first example is representative of our algorithm in the special case of Kummer extension. More precisely, we let $q = 256$ and consider the finite field $\mathbb{F}_{q^{2k}}$, with $k = q - 1$.

In this computation, the most costly part is the linear algebra step for computing the discrete logarithms of approximately 2^{22} quadratic polynomials. This is decomposed into 129 independent linear systems, one containing 2^{14} elements and 128 with 2^{15} elements. On average, these system contain 128 non-zero coefficients per line.

An initial phase of continued fractions reduced the problem to computed logarithms of polynomials of degree at most 29. The classical descent step was used to reduce this down to degree 12. The bilinear system approach permitted to conclude the computation.

The total cost of the individual logarithm was approximately one half of the cost of linear algebra. However, by using improved parameter choices (as in the next computation), it would be possible to reduce this by a large factor.

8.2 A Twisted Kummer Extension $\mathbb{F}_{256^{3\cdot 257}}$

This second example is interesting because it shows that pairing-based cryptography over $\mathbb{F}_{2^{257}}$ cannot be secure. However, it is too specific to be representative, indeed, it crucially relies on the fact that $\mathbb{F}_{256^3} = \mathbb{F}_{64^4}$.

The main specificity of this computation is a descent strategy, similar to the one presented in [6], that allows a descent from polynomials of degree 2 to polynomials of degree 1. This requires 3 conditions, the use of a Kummer or twisted Kummer extension, the replacement of the field of coefficients $\mathbb{F}_{q^2}$ by $\mathbb{F}_{q^3}$ and the use of two different polynomials to generate the systematic side of relations. Namely, we used both $X^{256} + X$ and $(X^{64} + X)^4$.

As a direct consequence, the costly phase of generating quadratic polynomials as a whole is removed. Thus, the computation becomes dominated by the descent phase. Compared to the previous computation, this was largely optimized. Indeed, the cost of the descent in this computation is about 1/10 of the descent in the previous example.

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).

Appendix

A Alternative Polynomials

Throughout the paper, we used the polynomial $X^q - X$ as our starting point. However, it is also possible to use other polynomials for this purpose. In order to be usable in our algorithm, a polynomial needs to satisfy two basic properties. First, it should factor into linear factors over a small degree extension of $\mathbb{F}_q$. Second, it should be possible to write it as a low degree polynomial in X and X^q.

Two possible alternative polynomials are $X^{q+1} - 1$ and $X^{q+1} + 1$ which factor into linear terms over $\mathbb{F}_{q^2}$. Another possibility is to use $X^{q+1} - X + 1$ or $X^{q+1} + X + 1$ which factor into linear terms over $\mathbb{F}_{q^3}$. For example, let us this factorization in the case of $X^{q+1} - X + 1$. Let x denote a root of this polynomial in $\overline{\mathbb{F}}_q$. It is clear that x satisfies:

$$x^q = \frac{x-1}{x}.$$

As a consequence:

$$x^{q^2} = \frac{x^q - 1}{x^q} = \frac{-1}{x-1},$$

and

$$x^{q^3} = \frac{-1}{x^q - 1} = x.$$

Thus x belongs to $\mathbb{F}_{q^3}$. The polynomials $X^{q+1} \pm X + 1$ are very closely related to the discrete logarithm approach proposed in [5].

A.1 Equivalence of Using the Alternative Polynomials

Assume that we are working with a subfield $\mathbb{F}_q$ of characteristic q. Define v to be a root of $X^{q+1} - 1$ in $\mathbb{F}_{q^2}$. Consider now the homography given by the quadruple $(a, b, c, d) = (v, 1, 1, v)$. It is easy to check that the image of $X^q - X$ by this homography is:

$$\begin{gathered}(ca^q - ac^q)X^{q+1} + (da^q - bc^q)X^q + (cb^q - ad^q)X + (db^q - bd^q) \equiv \\ (v^q - v)X^{q+1} + (v^{q+1} - 1)X^q + (1 - v^{q+1})X + (v - v^q) \equiv \\ (v^q - v)(X^{q+1} - 1).\end{gathered}$$

Up to a multiplicative constant, this yields the polynomial $X^{q+1} - 1$.

Similarly, if v denotes a root of $X^{q+1}+1$ in $\mathbb{F}_{q^2}$, consider the homography induced by the quadruple $(a,b,c,d)=(v,-1,1,v)$. The image of X^q-X is:

$$\begin{gathered}(ca^q-ac^q)X^{q+1}+(da^q-bc^q)X^q+(cb^q-ad^q)X+(db^q-bd^q)\equiv\\ (v^q-v)X^{q+1}+(v^{q+1}+1)X^q+(-1-v^{q+1})X+(-v+v^q)\equiv\\ (v^q-v)(X^{q+1}+1).\end{gathered}$$

As a consequence, the polynomials $X^{q+1}\pm 1$ can be obtained by applying a well-chosen homography to X^q-X. Thus, they do not generate any extra multiplicative relations in the finite field.

Similarly, the use of $X^{q+1}\pm X+1$ is equivalent to the use of X^q-X when taking coefficients in $\mathbb{F}_{q^3}$. To see that, define v to be a root of $X^{q+1}-X+1$ in $\mathbb{F}_{q^3}$. Consider the homography given by $(a,b,c,d)=(v,v-1,1,v)$. We see that after applying the homography, X^q-X becomes:

$$\begin{gathered}(ca^q-ac^q)X^{q+1}+(da^q-bc^q)X^q+(cb^q-ad^q)X+(db^q-bd^q)\equiv\\ (v^q-v)X^{q+1}+(v^{q+1}-v+1)X^q+(v^q-1-v^{q+1})X+(v^{q+1}-v-v^{q+1}+v^q)\equiv\\ (v^q-v)(X^{q+1}+X+1).\end{gathered}$$

Finally, with v a root of $X^{q+1}+X+1$ in $\mathbb{F}_{q^3}$ and the homography given by $(a,b,c,d)=(v,-v-1,1,v)$, we find after applying the homography:

$$\begin{gathered}(ca^q-ac^q)X^{q+1}+(da^q-bc^q)X^q+(cb^q-ad^q)X+(db^q-bd^q)\equiv\\ (v^q-v)X^{q+1}+(v^{q+1}+v+1)X^q+(-v^q-1-v^{q+1})X+(-v^{q+1}-v+v^{q+1}+v^q)\\ \equiv\\ (v^q-v)(X^{q+1}-X+1).\end{gathered}$$

As a consequence, we see that the four natural alternative polynomials that can be used with coefficients in $\mathbb{F}_{q^2}$ or $\mathbb{F}_{q^3}$ turn out to be equivalent to the use of X^q-X.

B Evidence for the Existence of the h_0 and h_1 Polynomials

Since our algorithm replies on the existence of low degree polynomials h_0 and h_1 such that $h_1(X)\cdot X^q-h_0(X)$ has a factor of degree k, it is important to study this heuristic hypothesis in more details.

In this appendix, we give some practical evidence for the existence of such polynomials in some practically interesting case. Assume that we wish to compute discrete logarithm in $\mathbb{F}_{2^p}$ for a prime p in the interval $[2^{10},2^{11}]$. We expect this to be achievable by embedding the finite field in $\mathbb{F}_{2^{11p}}$, i.e. by taking $q=2^{11}$. We define the finite field $\mathbb{F}_q$ as $\mathbb{F}_2[a]$, with $a^{11}+a^2+1=0$, and search for good polynomials h_0 and h_1 with coefficient in $\mathbb{F}_q$.

The result of this search is given in Table 1. It shows that all of the desired extension fields can be represented with polynomials h_0 and h_1 of degree 2.

Table 1. Representation of $\mathbb{F}_{q^p}$ by $X^q = h_0(X)/h_1(X)$ for $q = 2^{11}$

Extension degree	h_0	h_1	Extension degree	h_0	h_1
1031	$X^2 + a^{1555}\,X + a^{148}$	$X^2 + a^{1962}\,X + a^{1465}$	1033	$X^2 + a^{277}\,X + a^{702}$	$X^2 + a^{131}\,X + a^{1619}$
1039	$X^2 + a^{1161}\,X + a^{498}$	$X^2 + a^{1519}\,X + a^{1482}$	1049	$X^2 + a^{1768}\,X + a^{709}$	$X^2 + a^{131}\,X + a^{283}$
1051	$X^2 + a^{1967}\,X + a^{1919}$	$X^2 + a^{304}\,X + a^{272}$	1061	$X^2 + a^{638}\,X + a^{1905}$	$X^2 + a^{347}\,X + a^{651}$
1063	$X^2 + a^{1079}\,X + a^{525}$	$X^2 + a^{904}\,X + a^{2029}$	1069	$X^2 + a^{1050}\,X + a^{1725}$	$X^2 + a^{1842}\,X + a^{1551}$
1087	$X^2 + a^{421}\,X + a^{1405}$	$X^2 + a^{1404}\,X + a^{901}$	1091	$X^2 + a^{609}\,X + a^{1744}$	$X^2 + a^{1945}\,X + a^{781}$
1093	$X^2 + a^{608}\,X + a^{468}$	$X^2 + a^{342}\,X + a^{1200}$	1097	$X^2 + a^{1603}\,X + a^{452}$	$X^2 + a^{1910}\,X + a^{1892}$
1103	$X^2 + a^{155}\,X + a^{1694}$	$X^2 + a^{732}\,X + a^{779}$	1109	$X^2 + a^{414}\,X + a^{612}$	$X^2 + a^{656}\,X + a^{1029}$
1117	$X^2 + a^{409}\,X + a^{1303}$	$X^2 + a^{1591}\,X + a^{1159}$	1123	$X^2 + a^{46}\,X + a^{1131}$	$X^2 + a^{1615}\,X + a^{1379}$
1129	$X^2 + a^{194}\,X + a^{315}$	$X^2 + a^{1379}\,X + a^{1184}$	1151	$X^2 + a^{394}\,X + a^{391}$	$X^2 + a^{1305}\,X + a^{125}$
1153	$X^2 + a^{1673}\,X + a^{171}$	$X^2 + a^{870}\,X + a^{302}$	1163	$X^2 + a^{694}\,X + a^{1368}$	$X^2 + a^{220}\,X + a^{24}$
1171	$X^2 + a^{771}\,X + a^{1996}$	$X^2 + a^{306}\,X + a^{805}$	1181	$X^2 + a^{506}\,X + a^{2018}$	$X^2 + a^{326}\,X + a^{1698}$
1187	$X^2 + a^{1351}\,X + a^{1709}$	$X^2 + a^{1810}\,X + a^{1518}$	1193	$X^2 + a^{845}\,X + a^{42}$	$X^2 + a^{572}\,X + a^{900}$
1201	$X^2 + a^{1053}\,X + a^{175}$	$X^2 + a^{734}\,X + a^{1402}$	1213	$X^2 + a^{1562}\,X + a^{1541}$	$X^2 + a^{597}\,X + a^{704}$
1217	$X^2 + a^{715}\,X + a^{1251}$	$X^2 + a^{1085}\,X + a^{147}$	1223	$X^2 + a^{807}\,X + a^{1818}$	$X^2 + a^{599}\,X + a^{162}$
1229	$X^2 + a^{397}\,X + a^{1837}$	$X^2 + a^{823}\,X + a^{245}$	1231	$X^2 + a^{1750}\,X + a^{356}$	$X^2 + a^{59}\,X + a^{724}$
1237	$X^2 + a^{572}\,X + a^{922}$	$X^2 + a^{1784} * X + a^{2037}$	1249	$X^2 + a^{673}\,X + a^{902}$	$X^2 + a^{43}\,X + a^{877}$
1259	$X^2 + a^{1700}\,X + a^{1480}$	$X^2 + a^{1780}\,X + a^{1750}$	1277	$X^2 + a^{1380}\,X + a^{1484}$	$X^2 + a^{1861}\,X + a^{538}$
1279	$X^2 + a^{431}\,X + a^{1433}$	$X^2 + a^{1695}\,X + a^{438}$	1283	$X^2 + a^{493}\,X + a^{208}$	$X^2 + a^{85}\,X + a^{1672}$
1289	$X^2 + a^{1934}\,X + a^{1863}$	$X^2 + a^{1273}\,X + a^{1829}$	1291	$X^2 + a^{375}\,X + a^{524}$	$X^2 + a^{1236}\,X + a^{1945}$
1297	$X^2 + a^{1921}\,X + a^{1736}$	$X^2 + a^{598}\,X + a^{1530}$	1301	$X^2 + a^{1029}\,X + a^{478}$	$X^2 + a^{1434}\,X + a^{1418}$
1303	$X^2 + a^{1194}\,X + a^{1801}$	$X^2 + a^{208}\,X + a^{1592}$	1307	$X^2 + a^{1754}\,X + a^{626}$	$X^2 + a^{235}\,X + a^{979}$
1319	$X^2 + a^{1437}\,X + a^{282}$	$X^2 + a^{148}\,X + a^{744}$	1321	$X^2 + a^{982}\,X + a^{1089}$	$X^2 + a^{1632}\,X + a^{1598}$

Table 1. (*Continued*)

Extension degree	h_0	h_1	Extension degree	h_0	h_1
1327	$X^2 + a^{1455} X + a^{181}$	$X^2 + a^{508} X + a^{373}$	1361	$X^2 + a^{1451} X + a^{882}$	$X^2 + a^{1035} X + a^{634}$
1367	$X^2 + a^{331} X + a^{198}$	$X^2 + a^{1167} X + a^{1818}$	1373	$X^2 + a^{459} X + a^{1461}$	$X^2 + a^{946} X + a^{957}$
1381	$X^2 + a^{45} X + a^{1524}$	$X^2 + a^{1816} X + a^{766}$	1399	$X^2 + a^{684} X + a^{1574}$	$X^2 + a^{580} X + a^{1611}$
1409	$X^2 + a^{1439} X + a^{454}$	$X^2 + a^{1599} X + a^{1039}$	1423	$X^2 + a^{792} X + a^{1028}$	$X^2 + a^{940} X + a^{1662}$
1427	$X^2 + a^{345} X + a^{908}$	$X^2 + a^{1392} X + a^{864}$	1429	$X^2 + a^{667} X + a^{1656}$	$X^2 + a^{1867} X + a^{830}$
1433	$X^2 + a^{219} X + a^{362}$	$X^2 + a^{141} X + a^{1881}$	1439	$X^2 + a^{1417} X + a^{1761}$	$X^2 + a^{1224} X + a^{766}$
1447	$X^2 + a^{994} X + a^{1216}$	$X^2 + a^{15} X + a^{756}$	1451	$X^2 + a^{718} X + a^{766}$	$X^2 + a^{509} X + a^{702}$
1453	$X^2 + a^{1180} X + a^{129}$	$X^2 + a^{130} X + a^{1659}$	1459	$X^2 + a^{619} X + a^{782}$	$X^2 + a^{1423} X + a^{793}$
1471	$X^2 + a^{757} X + a^{210}$	$X^2 + a^{1192} X + a^{1976}$	1481	$X^2 + a^{1880} X + a^{882}$	$X^2 + a^{773} X + a^{339}$
1483	$X^2 + a^{670} X + a^{20}$	$X^2 + a^{24} X + a^{1514}$	1487	$X^2 + a^{1972} X + a^{1964}$	$X^2 + a^{1370} X + a^{528}$
1489	$X^2 + a^{1501} X + a^{116}$	$X^2 + a^{866} X + a^{694}$	1493	$X^2 + a^{1957} X + a^{987}$	$X^2 + a^{979} X + a^{781}$
1499	$X^2 + a^{1456} X + a^{1644}$	$X^2 + a^{1479} X + a^{600}$	1511	$X^2 + a^{279} X + a^{1360}$	$X^2 + a^{591} X + a^{1944}$
1523	$X^2 + a^{810} X + a^{25}$	$X^2 + a^{1924} X + a^{927}$	1531	$X^2 + a^{1415} X + a^{632}$	$X^2 + a^{1575} X + a^{911}$
1543	$X^2 + a^{1957} X + a^{1106}$	$X^2 + a^{1098} X + a^{1111}$	1549	$X^2 + a^{140} X + a^{498}$	$X^2 + a^{513} X + a^{1876}$
1553	$X^2 + a^{1109} X + a^{883}$	$X^2 + a^{1256} X + a^{524}$	1559	$X^2 + a^{485} X + a^{1312}$	$X^2 + a^{1102} X + a^{847}$
1567	$X^2 + a^{908} X + a^{128}$	$X^2 + a^{188} X + a^{194}$	1571	$X^2 + a^{29} X + a^{1916}$	$X^2 + a^{1825} X + a^{1266}$
1579	$X^2 + a^{953} X + a^{1192}$	$X^2 + a^{1113} X + a^{1334}$	1583	$X^2 + a^{792} X + a^{1459}$	$X^2 + a^{1115} X + a^{645}$
1597	$X^2 + a^{874} X + a^{1697}$	$X^2 + a^{387} X + a^{763}$	1601	$X^2 + a^{138} X + a^{1728}$	$X^2 + a^{1623} X + a^{961}$
1607	$X^2 + a^{737} X + a^{119}$	$X^2 + a^{1858} X + a^{1788}$	1609	$X^2 + a^{1641} X + a^{355}$	$X^2 + a^{1823} X + a^{963}$
1613	$X^2 + a^{801} X + a^{730}$	$X^2 + a^{193} X + a^{292}$	1619	$X^2 + a^{1715} X + a^{167}$	$X^2 + a^{510} X + a^{1166}$
1621	$X^2 + a^{1359} X + a^{745}$	$X^2 + a^{1157} X + a^{145}$	1627	$X^2 + a^{1560} X + a^{1074}$	$X^2 + a^{1631} X + a^{1624}$
1637	$X^2 + a^{575} X + a^{1741}$	$X^2 + a^{1620} X + a^{110}$	1657	$X^2 + a^{1727} X + a^{1064}$	$X^2 + a^{1968} X + a^{1714}$
1663	$X^2 + a^{960} X + a^{270}$	$X^2 + a^{744} X + a^{157}$	1667	$X^2 + a^{176} X + a^{536}$	$X^2 + a^{1208} X + a^{1919}$
1669	$X^2 + a^{229} X + a^{407}$	$X^2 + a^{1723} X + a^{1999}$	1693	$X^2 + a^{73} X + a^{642}$	$X^2 + a^{889} X + a^{489}$

Table 1. (*Continued*)

Extension degree	h_0	h_1	Extension degree	h_0	h_1
1697	$X^2 + a^{441} X + a^{722}$	$X^2 + a^{1454} X + a^{1566}$	1699	$X^2 + a^{387} X + a^{1300}$	$X^2 + a^{44} X + a^{684}$
1709	$X^2 + a^{1475} X + a^{1582}$	$X^2 + a^{63} X + a^{1779}$	1721	$X^2 + a^{1051} X + a^{846}$	$X^2 + a^{1536} X + a^{1506}$
1723	$X^2 + a^{1493} X + a^{1551}$	$X^2 + a^{1293} X + a^{1781}$	1733	$X^2 + a^{1536} X + a^{708}$	$X^2 + a^{836} X + a^{1518}$
1741	$X^2 + a^{1215} X + a^{455}$	$X^2 + a^{2013} X + a^{1400}$	1747	$X^2 + a^{978} X + a^{1676}$	$X^2 + a^{1444} X + a^{1102}$
1753	$X^2 + a^{450} X + a^{1685}$	$X^2 + a^{392} X + a^{136}$	1759	$X^2 + a^{1010} X + a^{1438}$	$X^2 + a^{1215} X + a^{63}$
1777	$X^2 + a^{1293} X + a^{249}$	$X^2 + a^{569} X + a^{554}$	1783	$X^2 + a^{150} X + a^{1608}$	$X^2 + a^{1185} X + a^{1061}$
1787	$X^2 + a^{1563} X + 1$	$X^2 + a^{1766} X + a^{1790}$	1789	$X^2 + a^{1435} X + a^{1084}$	$X^2 + a^{264} X + a^{770}$
1801	$X^2 + a^{1713} X + a^{678}$	$X^2 + a^{1656} X + a^{1626}$	1811	$X^2 + a^{1809} X + a^{2036}$	$X^2 + a^{1859} X + a^{525}$
1823	$X^2 + a^{659} X + a^{567}$	$X^2 + a^{147} X + a^{962}$	1831	$X^2 + a^{1384} X + a^{170}$	$X^2 + a^{550} X + a^{2035}$
1847	$X^2 + a^{885} X + a^{964}$	$X^2 + a^{701} X + a^{1221}$	1861	$X^2 + a^{1932} X + a^{1701}$	$X^2 + a^{158} X + a^{1250}$
1867	$X^2 + a^{1363} X + a^{1836}$	$X^2 + a^{307} X + a^{735}$	1871	$X^2 + a^{749} X + a^{1955}$	$X^2 + a^{499} X + a^{166}$
1873	$X^2 + a^{757} X + a^{200}$	$X^2 + a^{971} X + a^{601}$	1877	$X^2 + a^{758} X + a^{500}$	$X^2 + a^{943} X + a^{1832}$
1879	$X^2 + a^{289} X + a^{1359}$	$X^2 + a^{913} X + a^{840}$	1889	$X^2 + a^{1076} X + a^{1002}$	$X^2 + a^{1431} X + a^{476}$
1901	$X^2 + a^{752} X + a^{1060}$	$X^2 + a^{269} X + a^{1793}$	1907	$X^2 + a^{1954} X + a^{1856}$	$X^2 + a^{255} X + a^{316}$
1913	$X^2 + a^{1142} X + a^{578}$	$X^2 + a^{1118} X + a^{1052}$	1931	$X^2 + a^{1529} X + a^{777}$	$X^2 + a^{1631} X + a^{285}$
1933	$X^2 + a^{600} X + a^{509}$	$X^2 + a^{1477} X + a^{598}$	1949	$X^2 + a^{839} X + a^{1766}$	$X^2 + a^{1232} X + a^{226}$
1951	$X^2 + a^{1016} X + a^{1143}$	$X^2 + a^{1624} X + a^{1871}$	1973	$X^2 + a^{722} X + a^{769}$	$X^2 + a^{834} X + a^{1277}$
1979	$X^2 + a^{1007} X + a^{1464}$	$X^2 + a^{966} X + a^{912}$	1987	$X^2 + a^{1002} X + a^{682}$	$X^2 + a^{1255} X + a^{1006}$
1993	$X^2 + a^{709} X + a^{1676}$	$X^2 + a^{638} X + a^{957}$	1997	$X^2 + a^{1653} X + a^{1899}$	$X^2 + a^{29} X + a^{867}$
1999	$X^2 + a^{104} X + a^{1482}$	$X^2 + a^{1019} X + a^{649}$	2003	$X^2 + a^{328} X + a^{701}$	$X^2 + a^{554} X + a^{176}$
2011	$X^2 + a^{1510} X + a^{1241}$	$X^2 + a^{1524} X + a^{741}$	2017	$X^2 + a^{1572} X + a^{1645}$	$X^2 + a^{814} X + a^{298}$
2027	$X^2 + a^{1878} X + a^{1243}$	$X^2 + a^{1474} X + a^{1124}$	2029	$X^2 + a^{1502} X + a^{1998}$	$X^2 + a^{982} X + a^{721}$
2039	$X^2 + a^{1871} X + a^{1848}$	$X^2 + a^{1346} X + a^{1272}$			

References

1. Adleman, L.M., Huang, M.-D.A.: Function field sieve method for discrete logarithms over finite fields. Inf. Comput. **151**, 5–16 (1999). (Academic Press)
2. Barbulescu, R., Bouvier, C., Detrey, J., Gaudry, P., Jeljeli, H., Thomé, E., Videau, M., Zimmermann, P.: Discrete logarithm in GF(2^{809}) with FFS. IACR Cryptol. ePrint Arch. **2013**, 197 (2013)
3. Coppersmith, D.: Fast evaluation of logarithms in fields of characteristic two. IEEE Trans. Inf. Theor. **IT-30**(4), 587–594 (1984)
4. Faugère, J.-C., Din, M.S.E., Spaenlehauer, P.-J.: Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): algorithms and complexity. J. Symbolic Comput. **46**(4), 406–437 (2011)
5. Göloglu, F., Granger, R., McGuire, G., Zumbrägel, J.: On the function field sieve and the impact of higher splitting probabilities: application to discrete logarithms in 2^{1971}. IACR Cryptol. ePrint Arch. **2013**, 74 (2013)
6. Göloglu, F., Granger, R., McGuire, G., Zumbrägel, J.: Solving a 6120-bit DLP on a desktop computer. IACR Cryptol. ePrint Arch. **2013**, 306 (2013)
7. Huang, M.-D., Narayanan, A.K.: Finding primitive elements in finite fields of small characteristic. CoRR abs/1304.1206 (2013)
8. Joux, A.: Discrete logarithms in GF(2^{4080}). NMBRTHRY list, March 2013
9. Joux, A.: Discrete logarithms in GF(2^{6168}) = GF($(2^{257})^{24}$). NMBRTHRY list, May 2013
10. Joux, A.: Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 177–193. Springer, Heidelberg (2013)
11. Joux, A., Lercier, R.: The function field sieve in the medium prime case. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 254–270. Springer, Heidelberg (2006)
12. Joux, A., Lercier, R., Smart, N.P., Vercauteren, F.: The number field sieve in the medium prime case. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 326–344. Springer, Heidelberg (2006)
13. Panario, D., Gourdon, X., Flajolet, P.: An analytic approach to smooth polynomials over finite fields. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 226–236. Springer, Heidelberg (1998)
14. Semaev, I.: An algorithm for evaluation of discrete logarithms in some nonprime finite fields. Math. Comput. **67**, 1679–1689 (1998)
15. Spaenlehauer, P.-J.: Solving multi-homogeneous and determinantal systems Algorithms - Complexity - Applications. Ph.D. thesis, Université Pierre et Marie Curie (UPMC) (2012)