Solving a 6120-bit DLP on a Desktop Computer

Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel(✉)

Complex and Adaptive Systems Laboratory, School of Mathematical Sciences,
University College Dublin, Dublin, Ireland
{farukgologlu,robbiegranger}@gmail.com,
{gary.mcguire,jens.zumbragel}@ucd.ie

Abstract. In this paper we show how some recent ideas regarding the discrete logarithm problem (DLP) in finite fields of small characteristic may be applied to compute logarithms in some very large fields extremely efficiently. By combining the polynomial time relation generation from the authors' CRYPTO 2013 paper, an improved degree two elimination technique, and an analogue of Joux's recent small-degree elimination method, we solved a DLP in the record-sized finite field of 2^{6120} elements, using just a single core-month. Relative to the previous record set by Joux in the field of 2^{4080} elements, this represents a 50 % increase in the bitlength, using just 5 % of the core-hours. We also show that for the fields considered, the parameters for Joux's $L_Q(1/4 + o(1))$ algorithm may be optimised to produce an $L_Q(1/4)$ algorithm.

Keywords: Discrete logarithm problem · Binary finite fields

1 Introduction

The understanding of the hardness of the DLP in the multiplicative group of finite extension fields could be said to be undergoing a mini-revolution. It began with Joux's 2012 paper in which he introduced a method of relation generation dubbed 'pinpointing', which reduces the time required to obtain the logarithms of the elements of the factor base [11]. For medium-sized base fields, this technique has heuristic complexity as low as $L_Q(1/3, 2/3^{2/3}) \approx L_Q(1/3, 0.961)$[1], where

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

and Q is the cardinality of the finite field. This improves upon the previous best by Joux and Lercier [17] of $L_Q(1/3, 3^{1/3}) \approx L_Q(1/3, 1.442)$. To demonstrate the practicality of this approach, Joux solved two example DLPs in fields of bitlength 1175 and 1425 respectively, both with prime base fields.

Research supported by the Claude Shannon Institute, Science Foundation Ireland Grant 06/MI/006. The fourth author was in addition supported by SFI Grant 08/IN.1/I1950.

[1] On foot of recent communications [13], the complexity may in fact be $L_Q(1/3, 2^{1/3})$.

T. Lange, K. Lauter, and P. Lisoněk (Eds.): SAC 2013, LNCS 8282, pp. 136–152, 2014.
DOI: 10.1007/978-3-662-43414-7_7, © Springer-Verlag Berlin Heidelberg 2014

Soon afterwards the present authors showed that in the context of binary fields (and more generally small characteristic fields), finding relations for the factor base can be *polynomial time* in the size of the field [6]. By extending the basic idea to eliminate degree two elements during the descent phase, for medium-sized base fields a heuristic complexity as low as $L_Q(1/3, (4/9)^{1/3}) \approx L_Q(1/3, 0.763)$ was achieved; this approach was demonstrated via the solution of a DLP in the field $\mathbb{F}_{2^{1971}}$ [7], and in the field $\mathbb{F}_{2^{3164}}$.

After the initial publication of [6], Joux released a preprint [12] detailing an algorithm for solving the discrete logarithm problem for fields of the form $\mathbb{F}_{q^{2^n}}$, with $q = p^\ell$ and $n \approx q$, which was used in the solving of a DLP in $\mathbb{F}_{2^{1778}}$ [14], and later in $\mathbb{F}_{2^{4080}}$ [15]. This algorithm has heuristic complexity $L_Q(1/4 + o(1))$, and also has a heuristic polynomial time relation generation method, similar in principle to that in [6]. While the degree two element elimination in [6] is arguably superior, for other small degrees, Joux's elimination method is faster, resulting in the stated complexity. Joux's discrete logarithm computation in $\mathbb{F}_{2^{4080}}$ [15] required about 14,100 core-hours: 9,300 core-hours for the computation of the logarithms of all degree one and two elements; and 4,800 core-hours for the descent step, i.e., for computing the logarithm of an arbitrary element. For this computation, the field $\mathbb{F}_{2^{4080}}$ was represented as a degree 255 Kummer extension of $\mathbb{F}_{2^{16}}$, i.e., $\mathbb{F}_{(q^2)^{q-1}}$ with $q = 2^8$, as per [12]. The use of Kummer extensions (with extension degree either $q-1$ or $q+1$) gives a reduction in the size of the degree one and two factor base [11,12,17]; they are therefore preferable when it comes to setting record DLP computations.

The relation generation method in [6, Sect. 3.3] applies to larger base fields of the form $\mathbb{F}_{q^k}$ with $k \geq 3$ (rather than $k = 2$) and extension degrees up to $n \approx q\delta_1$ with $\delta_1 \geq 1$ a small integer. Hence the methods in this paper naturally apply to any extension degree. Note that this representation offers greater flexibility than Joux's (which can represent extension degrees up to $q + \delta_1'$) for essentially the same algorithmic cost, and may therefore provide a more practical DLP break when small base fields need to be embedded into larger ones in order to apply the attacks. However, here we choose to focus on Kummer extensions of degree $q \pm 1$, as these optimise the relation generation efficiency [6, Sect. 3.4], and linear algebra step. While the two DLP breaks in the fields $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$ contained therein did not fully exploit the above 'extreme' fields in which the extension degree is polynomially related to the size of the base field, thanks to Joux's fast small-degree elimination method, one can now do this more efficiently. Hence, with a view to solving the DLP in larger fields than before and in as short a time as possible, in this work we identify a family of fields for which the DLP is very easily solved, relative to other fields of a similar size. While this does not mean other fields of a similar size are infeasible to break, it requires more time in practice to find the logarithms of the factor base elements, with the complexities remaining the same.

One benefit of using base fields with $k \geq 3$ is that there is an efficient probabilistic elimination technique for degree two elements [6, Sect. 4.1]. For any fixed $k \geq 4$ the elimination probability very quickly tends to 1 for increasing q. In this paper we present an improved technique which allows one to find the logarithm

of degree two elements extremely fast, once the logarithms of all degree one elements are known. However, for $k = 3$ the elimination probability is $1/(2(\delta_1 - 1)!)$, or exactly $1/2$ for $\mathbb{F}_{2^{6120}} = \mathbb{F}_{(q^3)^{q-1}}$ with $q = 2^8$. Therefore the natural next choice is to set $k = 4$ and solve a DLP in $\mathbb{F}_{2^{8160}} = \mathbb{F}_{(q^4)^{q-1}}$. This would require solving a sparse linear system in $\approx 4.2 \cdot 10^6$ variables, and a slightly more costly descent step. Instead of carrying out this computation, we devised a technique for the 6120 bit case for which the elimination of each degree two element took only 0.03 s, and which required solving a much smaller linear system in 21,932 variables. This culminated in the resolution of a DLP in $\mathbb{F}_{2^{6120}}$ in under 750 core-hours [8], which represents a 50 % increase in bitlength over the previous record, whilst requiring just 5 % of the computation time.

We note that the solving of DLPs in $\mathbb{F}_{2^{6120}} = \mathbb{F}_{2^{24 \cdot 255}}$ renders insecure all pairing-based protocols based on supersingular curves of genus one and two over $\mathbb{F}_{2^{255}}$, since the correponding embedding degrees are 4 and 12 (in the best cases), respectively [1]. However, since 255 is not prime, such curves would not be recommended due to possible Weil descent attacks [5]. In any case, the Jacobians of the curves do not have prime or nearly prime order and so are not cryptographically interesting. As stated above, we could just as easily have solved the corresponding DLP with extension degree $q+1$ rather than $q-1$, i.e., with extension degree 257 rather than 255. However, since the full factorisation of $2^{6120} - 1$ is known, we were able to use a proven generator and so for completeness we chose to solve this case[2].

Since our break of the DLP in $\mathbb{F}_{2^{6120}}$ may be considered as a proof-of-concept implementation for our approach, at the time we were not overly concerned with the issue of complexity. Indeed, as the elimination times are reasonable and as just noted, comparable to Joux's elimination timings, further experimentation is needed to ascertain if the performance is comparable for larger systems. However, one basic difference between the two approaches is that the quadratic systems which arise when using our analogue of Joux's small-degree elimination method are not bilinear, and hence are not guaranteed to enjoy the same resolution complexity, as given in Spaenlehauer's thesis [25, Corollary 6.30]. Therefore, we can not currently argue that the heuristic complexity is the same. Nevertheless, we show that with a better choice of parameter and a tighter analysis, the final part of the descent in Joux's $L_Q(1/4 + o(1))$ algorithm may be improved to an $L_Q(1/4)$ algorithm, for the fields we consider, i.e., those for which the extension degree is polynomially related to the size of the basefield. Since the other phases of the algorithm have complexity $L_Q(1/4)$, or lower, the overall complexity for solving the DLP is $L_Q(1/4)$ as well.

[2] Forty days after the announcement of our full DLP break in $\mathbb{F}_{2^{6120}} = \mathbb{F}_{2^{24 \cdot 255}}$ [8] – and after the submission of this paper – Joux announced a break of the DLP in a 1843-bit subgroup of $\mathbb{F}^{\times}_{2^{6168}} = \mathbb{F}^{\times}_{2^{24 \cdot 257}}$, using a nearly identical degree two elimination technique and the same descent parameters, in under 550 core-hours [16]. Noting that the logarithms were not computed in the full multiplicative group and that this computation was performed on faster processors, it is clear that the number of our core-hours and Joux's are comparable. In this case too the corresponding Jacobians do not have prime or nearly prime order.

The remainder of the paper is organised as follows. Section 2 explains our field setup and algorithm in detail. Section 3 covers the other essential algorithms and issues regarding the computation. Section 4 gives the details of a discrete logarithm computation in $\mathbb{F}_{2^{6120}}$, while finally in Sect. 5 we briefly address the issue of complexity.

2 The Algorithm

The following describes the field setup and index calculus method that we use for our discrete logarithm computation.

2.1 Setup

We consider here Kummer extensions, which are our focus for efficiency reasons; the general case can be found in [6, Sect. 3.3] and is recalled in Sect. 5.

Let ℓ, k be positive integers, $q := 2^\ell$, and $n := q - 1$. We construct the finite field $\mathbb{F}_{(q^k)^n}$ of bit length $\ell kn = \ell k(q-1)$ in which we solve the DLP, as follows[3]. As stated in the introduction, the case $n := q + 1$ follows *mutatis mutandis*.

We express our base field $\mathbb{F}_{q^k}$ as a degree k extension of $\mathbb{F}_q$. Then we choose $\gamma \in \mathbb{F}_{q^k}$ such that the polynomial $X^n + \gamma$ is irreducible in $\mathbb{F}_{q^k}[X]$ and define $\mathbb{F}_{(q^k)^n}$ as the Kummer extension

$$\mathbb{F}_{q^k}(x) \cong \mathbb{F}_{q^k}[X]/\left((X^n + \gamma)\,\mathbb{F}_{q^k}[X]\right),$$

where x is a root of the polynomial $X^n + \gamma$ in $\mathbb{F}_{(q^k)^n}$. Note that a Kummer extension of degree n over $\mathbb{F}_{q^k}$ exists if and only if $n \mid q^k - 1$. Throughout the paper, the upper case letters $X, W, \ldots$ are used for indeterminates and the lower case letters $x, w, \ldots$ are reserved for finite fields elements that are roots of polynomials.

The following table displays the bit length ℓkn of the finite field $\mathbb{F}_{(q^k)^n}$ for various choices of the numbers ℓ and k.

$k \setminus \ell$	6	7	8	9
3	1134	2667	6120	13797
4	1512	3556	8160	18396
5	1890	4445	10200	22995
6	2268	5334	12240	27594

In Sect. 4, we will give the details of the discrete logarithm computation when $\ell kn = 6120$. The algorithm we explain in this section may be successfully applied to any of the above parameters with $k \geq 4$, whereas for $k = 3$ one would normally be required to precompute the logarithms of all degree two elements using a method analogous to Joux's [12]. However, for $k = 3$ and $\ell = 8$, precomputation can be avoided entirely; see Sect. 4.4.

[3] Our choice of representation of the finite field $\mathbb{F}_{(q^k)^n}$ will be advantageous for our method to solve the DLP. Note that it is a computationally easy problem to switch between two different representations of a finite field [22].

2.2 Factor Base and Automorphisms

The factor base we use consists of the elements in $\mathbb{F}_{(q^k)^n}$ which have degree one in the polynomial representation over $\mathbb{F}_{q^k}$, i.e., we consider the set $\{x + a \mid a \in \mathbb{F}_{q^k}\}$. As noted in [6,11,17], factor base preserving automorphisms of $\mathbb{F}_{(q^k)^n}$, which are provided by Kummer extensions, can be used to significantly reduce the number of variables involved in the linear algebra step. Indeed, the map $\sigma := \mathrm{Frob}^\ell : \alpha \to \alpha^q$ satisfies $\sigma(x) = \gamma x$ with $\gamma \in \mathbb{F}_{q^k}$, and thus preserves the factor base. Furthermore, for $\varphi := \sigma^k = \mathrm{Frob}^{\ell k} : \alpha \to \alpha^{q^k}$ we have $\varphi(x) = \mu x$ with $\mu \in \mathbb{F}_q$ a primitive n-th root of unity, and thus we find

$$(x+a)^{q^{kj+i}} = \sigma^{kj+i}(x+a) = \sigma^i(\varphi^j(x+a)) = \sigma^i(\mu^j x + a) = \mu^j \gamma^{e_i} x + a^{q^i} ,$$

where $e_0 = 0$ and $e_i = qe_{i-1} + 1$ for $1 \le i < k$; thus it follows that

$$\log\left(x + \frac{a^{q^i}}{\mu^j \gamma^{e_i}}\right) = q^{kj+i} \log(x+a)$$

for all $0 \le j < n$ and $0 \le i < k$.

The automorphism σ generates a group of order kn, which acts on the set of q^k factor base elements, thus dividing the factor base into about N orbits, where $N \approx \frac{q^k}{kn} \approx \frac{1}{k} q^{k-1}$ is the number of variables to consider.

2.3 Relation Generation

In order to generate relations between the factor base elements we use the method from [6, Sect. 3.1–4]. We exploit properties of polynomials of the form

$$F_B(X) := X^{q+1} + BX + B ,$$

which have been studied by Bluher [2] and Helleseth/Kholosha [10]. We recall in particular the following result of Bluher [2] (see also [6,10]):

Theorem 1. *The number of elements $B \in \mathbb{F}_{q^k}^\times$ such that the polynomial $F_B(X)$ splits completely over $\mathbb{F}_{q^k}$ equals*

$$\frac{q^{k-1}-1}{q^2-1} \quad \text{if } k \text{ odd}, \qquad \frac{q^{k-1}-q}{q^2-1} \quad \text{if } k \text{ even}.$$

Let $B \in \mathbb{F}_{q^k}^\times$ be an element such that $F_B(X)$ splits and denote its roots by μ_i, for $i = 1, \ldots, q+1$. For arbitrary $a, b \in \mathbb{F}_{q^k}$ (with $a^q \ne b$) there exists $c \in \mathbb{F}_{q^k}$ with $(a^q + b)^{q+1} = B\,(ab + c)^q$ and we then find that

$$f(X) := F_B\left(\frac{ab+c}{a^q+b} X + a\right) = X^{q+1} + aX^q + bX + c$$

and that $f(X)$ also splits over $\mathbb{F}_{q^k}$, with roots $\nu_i := \frac{ab+c}{a^q+b} \mu_i + a$.

Now by the definition of $\mathbb{F}_{(q^k)^n}$ we have $x^n = \gamma$ and thus $x^q = \gamma x$, with $\gamma \in \mathbb{F}_{q^k}$. Hence in $\mathbb{F}_{(q^k)^n}$ we have

$$f(x) = \gamma x^2 + a\gamma x + bx + c = \gamma(x^2 + (a + \tfrac{b}{\gamma})x + \tfrac{c}{\gamma}) = \gamma g(x)\,,$$

where $g(X) := X^2 + (a + \frac{b}{\gamma})X + \frac{c}{\gamma}$. Hence, if the polynomial $g(X)$ splits, i.e., if $g(X) = (X + \xi_1)(X + \xi_2)$, which heuristically occurs with probability $1/2$, then we find a relation of factor base elements, namely

$$\prod_{i=1}^{q+1}(x + \nu_i) = \gamma(x + \xi_1)(x + \xi_2)\,.$$

Such a relation corresponds to a linear relation between the logarithms of the factor base elements. Once we have found more than N relations we can solve the discrete logarithms of the factor base elements by means of linear algebra; see Sect. 3.3.

2.4 Individual Logarithms

After the logarithms of the factor base elements have been found, a general individual discrete logarithm can be computed, as is common, by a descent strategy. The basic idea of this method is trying to write an element, given by its polynomial representation over $\mathbb{F}_{q^k}$, as a product in $\mathbb{F}_{(q^k)^n}$ of factors represented by lower degree polynomials. By applying this principle recursively a descent tree is constructed, and one can eventually express a given target element by a product of factor base elements, thus solving the DLP.

While for large degree polynomials it is relatively easy to find an expression involving lower degree polynomials by a standard approach, this method becomes increasingly less efficient as the degree becomes smaller. In addition, the number of small degree polynomials in the descent tree grows significantly with lower degree. We therefore propose new methods for degree 2 elimination and small degree descent, which are inspired by the recent works [6] and [12] respectively.

Degree 2 Elimination. Given a polynomial $Q(X) := X^2 + q_1X + q_0 \in \mathbb{F}_{q^k}[X]$ we aim at expressing the corresponding finite field element $Q(x) \in \mathbb{F}_{(q^k)^n}$ as a product of factor base elements. In essence, what we do is just the reverse of the degree one relation generation, with the polynomial $g(X)$ set to be $Q(X)$.

In particular, we compute – when possible – $a, b, c \in \mathbb{F}_{q^k}$ such that, up to a multiplicative constant in $\mathbb{F}_{q^k}^\times$, $Q(x) = x^2 + q_1x + q_0$ equals $x^{q+1} + ax^q + bx + c$ where the polynomial $X^{q+1} + aX^q + bX + c$ splits into linear factors (cf. [6, Sect. 4.1]).

As $x^n = \gamma$ holds, we have $x^{q+1} + ax^q + bx + c = \gamma(x^2 + (a + \frac{b}{\gamma})x + \frac{c}{\gamma})$ and comparing coefficients we find $\gamma q_0 = c$ and $\gamma q_1 = \gamma a + b$. Now letting $B \in \mathbb{F}_{q^k}^\times$

be an element satisfying the splitting property of Theorem 1 and combining the previous equations with $(a^q + b)^{q+1} = B\,(ab + c)^q$ we arrive at the condition

$$(a^q + \gamma a + \gamma q_1)^{q+1} + B(\gamma a^2 + \gamma q_1 a + \gamma q_0)^q = 0\,.$$

Considering $\mathbb{F}_{q^k}$ as a degree k extension over $\mathbb{F}_q$ this equation gives a quadratic system in the k $\mathbb{F}_q$-components of a, which can be solved very fast by a Gröbner basis method.

Heuristically, for each of the above B's the probability of success of this method, i.e., when an $a \in \mathbb{F}_{q^k}$ as above exists, is 1/2. Note that if $k = 3$ there is just one single B in the context of Theorem 1, and so this direct method fails in half of the cases. However, as noted earlier, this issue can be resolved under certain circumstances, e.g., for $\ell = 8$; see Sect. 4.4.

Small Degree Descent. The following describes the Gröbner basis descent of Joux [12] applied in the context of the polynomials $F_B(X) = X^{q+1} + BX + B$ of Theorem 1. Let $f(X)$ and $g(X)$ be polynomials over $\mathbb{F}_{q^k}$ of degree δ_f and δ_g respectively. We substitute X by the rational function $\frac{f(X)}{g(X)}$ and thus find that the polynomial

$$P(X) := f(X)^{q+1} + Bf(X)\,g(X)^q + Bg(X)^{q+1}$$

factors into polynomials of degree at most $\delta = \max\{\delta_f, \delta_g\}$. Since $x^q = \gamma x$ holds in $\mathbb{F}_{(q^k)^n}$ the element $P(x)$ can also be represented by a polynomial of degree 2δ.

Now given a monic polynomial $Q(X) \in \mathbb{F}_{q^k}[X]$ of degree 2δ (resp. $2\delta - 1$) to be eliminated we consider the equation $P(x) = Q(x)$ (resp. $P(x) = (x + a)Q(x)$ with some random fixed $a \in \mathbb{F}_{q^k}$). It results as above in a quadratic system of $\mathbb{F}_q$-variables representing the coefficients of $f(X)$ and $g(X)$ in $\mathbb{F}_{q^k}$, and can be solved by a Gröbner basis algorithm. In order to minimise the number of variables involved we set $f(X)$ to be monic of degree $\delta_f = \delta$ and $g(X)$ of degree $\delta_g = \delta - 1$, resulting in $k\delta + k\delta = 2k\delta$ variables in $\mathbb{F}_q$. Since the number of equations to be satisfied equals $2k\delta$ as well, we find a solution of this system with good probability.

Large Degree Descent. This part of the descent is somewhat classical (see [17] for example), but includes the degree balancing technique described in [6, Sect. 4], which makes the descent far more rapid when the base field $\mathbb{F}_{q^k}$ is a degree k extension of a non-prime field. In the finite field $\mathbb{F}_{(q^k)^n}$ we let $y := x^q$ and $\bar{x} := x^{2^{\ell-a}}$ for some suitably chosen integer $1 < a < k$. Then $y = \bar{x}^{2^a}$ and $\bar{x} = (\frac{y}{\gamma})^{2^{\ell-a}}$ holds. Now for given $Q(X) \in \mathbb{F}_{q^k}[X]$ of degree d representing $Q(y)$ we consider the lattice

$$L := \left\{(w_0, w_1) : Q(X) \mid (\tfrac{X}{\gamma})^{2^{\ell-a}} w_0(X) + w_1(X)\right\} \subseteq \mathbb{F}_{q^k}[X]^2\,.$$

By Gaussian lattice reduction we find a basis (u_0, u_1), (v_0, v_1) of L of degree $\approx d/2$ and can thus generate lattice elements $(w_0, w_1) = r(u_0, u_1) + s(v_0, v_1)$ of

low degree. In $\mathbb{F}_{(q^k)^n}$ we then consider the equation

$$\bar{x}w_0(\bar{x}^{2^a}) + w_1(\bar{x}^{2^a}) = \bar{x}w_0(y) + w_1(y) = (\tfrac{y}{\gamma})^{2^{\ell-a}} w_0(y) + w_1(y)\,,$$

where the right-hand side is divisible by $Q(y)$ by construction, and a is chosen so as to make the degrees of both sides as close as possible. The descent is successful whenever a lattice element (w_0, w_1) is found such that the involved polynomials $Xw_0(X^{2^a}) + w_1(X^{2^a})$ and $\frac{1}{Q(x)}(X^{2^{\ell-a}} w_0(X) + \gamma^{2^{\ell-a}} w_1(X))$ are $(d-1)$-smooth, i.e., have only factors of degree less than d.

3 Other Essentials

In this section we give an explicit account of further basics required for a discrete logarithm computation.

3.1 Factorisation of the Group Order

The factorisation of the group order $|\mathbb{F}^{\times}_{(q^k)^n}| = 2^{\ell kn} - 1$ is of interest for several reasons. Firstly it indicates the difficulty of solving the associated DLP using the Pohlig-Hellman algorithm. Secondly it enables one to provably find a generator. Finally, it determines the small factors for which we apply Pollard's rho method, and the large factors for the linear algebra computation. Since the complexity of the Special Number Field Sieve [20] is much higher than the present DLP algorithms, it is unlikely that one can completely factorise $2^{\ell kn} - 1$ in cases of interest in a reasonable time. In these cases it is vital to at least know all the small prime factors of the group order, which can be accomplished using the Elliptic Curve Method [21] and the identity

$$2^{\ell kn} - 1 = \prod_{d \mid \ell kn} \Phi_d(2)\,,$$

where $\Phi_d \in \mathbb{Z}[x]$ denotes the d-th cyclotomic polynomial.

3.2 Pohlig-Hellman and Pollard's Rho Method

In order to compute a discrete logarithm in a group G of order m we can use any factorisation of $m = m_1 \cdot \ldots \cdot m_r$ into pairwise coprime factors m_i and compute the discrete log modulo each factor. Indeed, if we are to compute $z = \log_\alpha \beta$ it suffices to compute $\log_{\alpha^{c_i}} \beta^{c_i}$ with $c_i = m/m_i$, which determines $z \bmod m_i$. With the information of $z \bmod m_i$ for all i one easily determines $z \pmod m$ by the Chinese Remainder theorem.

For the small prime (power) factors of m we use Pollard's rho method to compute the discrete logarithm modulo each factor. Regarding the large factors of m we find it most efficient to combine them into a single product m_*, so that in the linear algebra step of the index calculus method we work over the ring $\mathbb{Z}_{m_*}$. Note that each iteration of the Lanczos method that we use for the linear algebra problem requires the inversion of a random element in $\mathbb{Z}_{m_*}$; this is the reason why we separate the small factors of the group order from the large ones.

3.3 Linear Algebra

The relation generation phase of the index calculus method produces linear relations among the logarithms of the factor base elements. As the factor base logs are also related by the automorphism group as explained in Sect. 2.2 the number N of variables is reduced and the linear relations will have coefficients being powers of 2. Once $M > N$ relations have been generated we have to find a nonzero solution vector for the linear system. To ensure that the matrix is of maximal rank $N-1$ we generate $M \approx N + 100$ relations. As noted earlier the number of variables N is expected to be about $\frac{q^k}{kn} \approx \frac{1}{k} q^{k-1}$.

We let B be the $M \times N$ matrix of the relations' coefficients, which is a matrix of constant row-weight $q+3$. We have to find a nonzero vector v of length N such that $Bv = 0$ modulo m_*, the product of the large prime factors of the group order m. A common approach in index calculus algorithms is to reduce the matrix size at this stage by using a structured Gaussian elimination (SGE) method. In our case, however, the matrix is not extremely sparse while its size is quite moderate, hence the expected benefit from SGE would be minimal and we refrained from this step.

We use the iterative Lanczos method [18,19] to solve the linear algebra problem, which we briefly describe here. Let $A = B^t B$, which is a symmetric $N \times N$ matrix. We let $v \in \mathbb{Z}_{m_*}^N$ be random, $w = Av$, and find a vector $x \in \mathbb{Z}_{m_*}^N$ such that $Ax = w$ holds (since $A(x-v) = 0$ we have thus found a kernel element). We compute the following iteration

$$\begin{aligned} w_0 = w, \quad v_0 = Aw_0, \qquad & w_1 = v_0 - \tfrac{(v_0,v_0)}{(v_0,w_0)} w_0 \\ v_i = Aw_i, \qquad & w_{i+1} = v_i - \tfrac{(v_i,v_i)}{(v_i,w_i)} w_i - \tfrac{(v_i,v_{i-1})}{(v_{i-1},w_{i-1})} w_{i-1} \end{aligned}$$

and stop once $(v_j, w_j) = 0$; if $w_j \neq 0$ the algorithm fails, otherwise we find the solution vector

$$x = \sum_{i=0}^{j-1} \frac{(w, w_i)}{(v_i, w_i)} w_i \, .$$

Performing the above iteration consists essentially of several matrix-vector products, scalar-vector multiplications, and vector-vector inner products. As the matrix is sparse and consists of entries being powers of 2 the matrix-vector products can be carried out quite efficiently. Therefore, the scalar multiplications and inner products consume a significant part of the computation time. We have used a way to reduce the number of inner products per iteration, as was suggested recently [23].

Indeed, using the A-orthogonality $(v_i, w_j) = w_i^t A w_j = 0$ for $i \neq j$ we find that

$$(v_i, v_{i-1}) = (v_i, w_i) \quad \text{and} \quad (w, w_{i+1}) = -\frac{(v_i, v_i)}{(v_i, w_i)}(w, w_i) - \frac{(v_i, v_{i-1})}{(v_{i-1}, w_{i-1})}(w, w_{i-1}).$$

Now at each iteration, given w_i we compute the matrix-vector product Bw_i and the inner product $a_i := (v_i, w_i) = (Bw_i, Bw_i)$, as well as $v_i = Aw_i = B^t(Bw_i)$

and $b_i := (v_i, v_i) = (Aw_i, Aw_i)$. We then have the simplified iteration

$$w_0 = w\,, \quad w_1 = v_0 - \frac{b_0}{a_0} w_0\,, \quad w_{i+1} = v_i - \frac{b_i}{a_i} w_i - \frac{a_i}{a_{i-1}} w_{i-1}$$

and the solution vector $x = \sum_{i=0}^{j-1} \frac{c_i}{a_i} w_i$, where $c_i := (w, w_i)$ can be computed by the iteration

$$c_0 = (w, w)\,, \quad c_1 = a_0 - \frac{b_0}{a_0} c_0\,, \quad c_{i+1} = -\frac{b_i}{a_i} c_i - \frac{a_i}{a_{i-1}} c_{i-1}\,.$$

We see that each iteration requires merely two matrix-vector products, three scalar multiplications, and two inner products.

3.4 Target Element

In order to set ourselves a DLP challenge we construct the 'random' target element $\beta \in \mathbb{F}_{(q^k)^n}$ using the binary digits expansion of the mathematical constant π. More precisely, considering the q^k-ary expansion

$$\pi = 3 + \sum_{i=1}^{\infty} c_i\, q^{-ki} \quad \text{with} \quad c_i \in S_{q^k} := \{0, 1, \ldots, q^k - 1\}$$

we use a bijection between the sets S_{q^k} and $\mathbb{F}_{q^k}$, which is defined by the mappings $\varphi_q : \mathbb{F}_q \to \{0, \ldots, q-1\}$: $\sum_{i=0}^{\ell-1} a_i t^i \mapsto \sum_{i=0}^{\ell-1} a_i 2^i$ and $\varphi : \mathbb{F}_{q^k} \to S_{q^k}$: $\sum_{j=0}^{k-1} b_j w^j \mapsto \sum_{j=0}^{k-1} \varphi_q(b_j) q^j$, and construct in this way the target element

$$\beta_\pi := \sum_{i=0}^{n-1} \varphi^{-1}(c_{i+1})\, x^i \in \mathbb{F}_{(q^k)^n}\,.$$

4 Discrete Logarithms in $\mathbb{F}_{2^{6120}}$

In this section we document the breaking of a DLP in the case $\ell = 8$ and $k = 3$, i.e., in $\mathbb{F}_{2^{6120}}$. The salient features of the computation are:

- The relation generation for degree one elements took 15 s[4].
- The corresponding linear algebra took 60.5 core-hours.
- In contrast to [12,15], we computed the logarithm of degree 2 irreducibles on the fly; each took on average 0.03 s.
- The descent was designed so as to significantly reduce the number of bottleneck (degree 6) eliminations. As a result, the individual logarithm phase took just under 689 core-hours.

[4] In our inital announcement [8] we stated a running time of 60 s for the relation generation. The reason for this higher running time was an unnecessary step of ordering the matrix entries, which we have discounted here.

4.1 Setup

We first defined $\mathbb{F}_{2^8}$ using the irreducible polynomial $T^8 + T^4 + T^3 + T + 1$. Letting t be a root of this polynomial, we defined $\mathbb{F}_{2^{24}}/\mathbb{F}_{2^8}$ using the irreducible polynomial $W^3 + t$. Letting w be a root of this polynomial, we finally defined $\mathbb{F}_{2^{6120}}/\mathbb{F}_{2^{24}}$ using the irreducible polynomial $X^{255} + w + 1$, where we denote a root of this polynomial by x.

We chose as a generator $g = x + w$, which has order $2^{6120} - 1$; this was proven via the prime factorisation of $2^{6120} - 1$, which is provided in [8]. As usual, the target element was set to be β_π as explained in Sect. 3.4.

4.2 Relation Generation

Our factor base is simply the set of degree one elements of $\mathbb{F}_{2^{6120}}/\mathbb{F}_{2^{24}}$. As detailed in Sect. 2.2, quotienting out by the action of the 8-th power of Frobenius produces 21,932 distinct orbits. To obtain relations, as explained in Sect. 2.3, we make essential use of the single polynomial $X^{257} + X + 1$, which splits completely over $\mathbb{F}_{2^{24}}$. In particular, letting $y := x^{256}$ so that $x = \frac{y}{w+1}$, the $\mathbb{F}_{2^{6120}}$ element $xy + ay + bx + c$ corresponds to $X^{257} + aX^{256} + bX + c$ on the one hand, and $\frac{X^2}{w+1} + aX + \frac{bX}{w+1} + c$ on the other. The first of these transforms to $X^{257} + X + 1$ if and only if $(a^{256} + b)^{257} = (ab + c)^{256}$. So for randomly chosen (a, b) we compute c and check whether the corresponding quadratic splits. If it does – which occurs with probability $1/2$ – we obtain a relation. Thanks to the simplicity of this approach, we collected 22,932 relations and wrote these to a matrix in 15 s using C++/NTL [24].

4.3 Linear Algebra

We took as our modulus the product of the largest 35 factors of $2^{6120} - 1$ listed in [8], which has bitlength 5121. We ran a parallelised C/GMP [9] implementation of Lanczos' algorithm on four of the Intel (Westmere) Xeon E5650 hex-core processors of ICHEC's SGI Altix ICE 8200EX Stokes cluster. This took 60.5 core-hours (just over 2.5 h wall time).

4.4 Individual Logarithm

Degree 2 Elimination. For computing the discrete logarithm of a degree two element $Q(x) = x^2 + q_1 x + q_0$ we try to equate $Q(x)$ with $x^{257} + ax^{256} + bx + c$, where $(a^{256} + b)^{257} = (ab + c)^{256}$. If this fails we apply the following strategy, making use of the fact that $\mathbb{F}_{2^{24}}$ can also be viewed as a field extension of $\mathbb{F}_{2^6}$. We consider $y = x^{256}$ and $\bar{x} = x^4$, so that $y = \bar{x}^{64}$ and $\bar{x} = (\frac{y}{\gamma})^4$ holds, and apply the large degree descent method to $\bar{Q}(X) := Q(\frac{X}{\gamma})$ (note that $\bar{Q}(y) = Q(x)$). Considering the lattice L (see Sect. 2.4) we construct a basis of the form $(X + u_0, u_1)$, $(v_0, X + v_1)$, where $u_0, u_1, v_0, v_1 \in \mathbb{F}_{2^{24}}$. Then for $s \in \mathbb{F}_{2^{24}}$ we have

lattice elements $(X + u_0 + sv_0, sX + u_1 + sv_1) \in L$. Now for each $B \in \mathbb{F}_{2^{24}}$ such that $X^{65} + BX + B$ splits, we solve for $s \in \mathbb{F}_{2^{24}}$ satisfying

$$(v_0 s^2 + (u_0 + v_1)s + u_1)^{64} = B\,(s^{64} + v_0 s + u_0)^{65}\,,$$

which can be expressed as a quadratic system in the $\mathbb{F}_{2^6}$-components of s, and thus solved by a Gröbner basis computation over $\mathbb{F}_{2^6}$. We then have an equation

$$\bar{x}^{65} + a\bar{x}^{64} + b\bar{x} + c = \tfrac{1}{\gamma^4}(y^5 + by^4 + a\gamma^4 y + c\gamma^4)$$

with $a = s$, $b = \gamma s + q_1$, and $c = \frac{q_0}{\gamma}$, where the left-hand side polynomial splits, while the right-hand side polynomial contains $\bar{Q}(X)$.

The polynomial $X^5 + bX^4 + a\gamma^4 X + c\gamma^4 = \bar{Q}(X)R(X)$ has the property that $R(X)$ always factors into a linear and an irreducible quadratic polynomial over $\mathbb{F}_{q^k}$. Indeed, by a result of Bluher [2, Theorem 4.3], for any $B \in \mathbb{F}_{2^{24}}$ and any $d \geq 1$, the number of roots in $\mathbb{F}_{2^{24d}}$ of the polynomial $F_B(X) = X^5 + BX + B$ equals either 0, 1, 2, or 5. Since $X^5 + bX^4 + a\gamma^4 X + c\gamma^4$ can be rewritten as $X^5 + BX + B$ via a linear transformation (except when $a\gamma^4 = b^4$), the same holds also regarding the $\mathbb{F}_{2^{24d}}$-roots of this polynomial. Now applying Bluher's result for $d = 1$ we see that $R(X)$ can not split into linear factors, and by Bluher's result for $d = 3$ we conclude that $R(X)$ can not be irreducible. Hence, $R(X)$ is the product of linear and a quadratic polynomial, which we call $Q'(X)$.

Now if $Q'(X)$ is resolvable by the direct method, we have successfully eliminated the original polynomial $Q(X)$. The number of B such that $X^{65} + BX + B$ splits over $\mathbb{F}_q$ equals 64, according to Theorem 1, and by experiment, for each one the success probability to find a resolvable polynomial $Q'(X)$ is about 0.4.

Performing the Descent. Using C++/NTL we first used continued fractions to express the target element β_π as a ratio of two 27-smooth polynomials, which took 10 core-hours, and then we applied the three different descent strategies as explained in Sect. 2.4.

We used the large degree descent strategy to express all of the featured polynomials using polynomials of degree 6 or less. This took a further 495 core-hours. While we could have performed this part of the descent more efficiently, as noted above we opted to find expressions which resulted in a relatively small number of degree 6 polynomials – which are the bottleneck eliminations for the subsequent descent – namely 326.

For degrees 6 down to 3 we used the analogue of Joux's small degree elimination method, based on the same polynomial that we used for relation generation, i.e., $X^{257} + X + 1$, rather than the polynomial $X^{256} + X$ that was used in [15], since the resulting performance was slightly better. Finally, we performed the degree 2 elimination as outlined above.

For convenience we coded the eliminations of polynomials of degrees 6 down to 2 in Magma [3] V2.16-12, using Faugere's F4 algorithm [4]. The total time for this part was just over 183.5 core-hours on a 2 GHz AMD Opteron computer.

For the logarithm modulo the cofactor of our modulus we used either linear search or Pollard's rho method, which took 20 min in total in C++/NTL. Thus the total time for the descent was just under 689 h.

Finally, we found[5] that $\beta_\pi = g^{\log}$, with log =

1385875983639786926254757112831231710092363615038969923664959317045177002801271780222348940986175813601314418350742563637306244268142932334742725215981661269579281168254431109654042538379388085954041110352380271077721788229392818734034519997318151400734817665137153584492793145567973524462468603179467501244756894744062749423560359365016740509334489092010298345222267322477718970832232172820515736450136036130423677827163618778179383743938243130190736247863876184140375416811202840446593831929074368525263920877243047754516312718252509681114514005027334043817696752552891273466393500982215708444003807885163324965838825224363819180082001670321863502451077513469795963146961536667161689514819480910600667301847667581377739443038754298308672054639181442568439117307472651461541934380416278336617397750571612363460962365668752512778430623299730444754865610622043569085684714712793837810385388188844637969899060760798432481272520208397058864360712136505751867074569485840723789169429253691408684171964795734810327114810217291628659735881740963899133056076778580339963617349055371503620247205157726607812088555054343310557665700142118756029406335757638504575030790870743765853044705204113202462922553757114575735552860602366993170394544793267182811289614232751427875694256905328332833440496355213025960008971925120366952988072940329645309596913770872045463489601327600955441059801982552454932024128315938919847881524179576919398171123661820636875299153651503611802144512343876568832561493559944050511495859691630753070266479560356836715895464485399551327261120349386559612918562034222476803870290784735209511603344725254750716806726236615872927203296061825120443121943571561392013409520378729752432544760815549370021229534159494072621372320998522983948384229076431913976732902383441830460409758599159285365304456971453176680449737096483324156185041.

4.5 Total Running Time

The total running time is $689 + 60.5 = 749.5$ core-hours. Note that most of the computation (all except the linear algebra part) was performed on a personal computer. On a modern quad-core PC, the total running time would be around a week.

[5] Magma verification code for this solution is available from [8].

5 Complexity Considerations

In this section we prove a tighter complexity result than that given in [12] for the new small-degree stage of the descent. As stated in Sect. 1, the systems arising from the small-degree elimination in Sect. 2.4 are quadratic, but not bilinear. As such, they do not necessarily enjoy the same resolution complexity as bilinear quadratic systems, as given by a theorem due to Spaenlehauer [25, Corollary 6.30]. However, if one instead reverts to using the polynomial $X^q - X$, then one can argue as follows.

Let the fields under consideration be $\mathbb{F}_{(q^k)^n}$, with $k \geq 3$ fixed, $n \approx q\delta_1$ and $\delta_1 \geq 1$ a small integer, as per the field representation described in [6, Sect. 3.3], and $q \to \infty$. This is achieved by finding a polynomial p_1 of degree δ_1 such that $p_1(X^q) - X \equiv 0 \pmod{I(X)}$, with $I(X)$ irreducible of degree n. By letting $x \in \mathbb{F}_{(q^k)^n}$ be a root of $I(X)$ and $y := x^q$, one also has $x = p_1(y)$, and therefore two related representations of $\mathbb{F}_{(q^k)^n}$.

For simplicity we assume $\delta_1 = 1$; the case $\delta_1 > 1$ can be treated similarly. The cardinality of $\mathbb{F}_{(q^k)^n}$ is $\approx q^{kq}$ and we have

$$\begin{aligned} L_{q^{kq}}(1/4, c) &= \exp\left((c + o(1))(kq \log q)^{1/4} (\log(kq \log q))^{3/4}\right) \\ &= \exp\left((ck^{1/4} + o(1))\, q^{1/4} \log q\right). \end{aligned} \tag{1}$$

We now recall Joux's elimination method. The final part of the descent starts with an element $Q(x)$ of degree $D \approx \alpha_1 q^{1/2}$ which is to be eliminated; here, α_1 is a constant that depends on the efficiency of the classical large-degree descent. For a parameter $1 < d < D/2$ yet to be optimised, we substitute $X = f(X)/g(X)$ into $X^q - X$ with $\deg(f) = d$ and $\deg(g) = D - d$, both with yet-to-be determined $\mathbb{F}_{q^k}$ coefficients. In this case one has the $\mathbb{F}_{(q^k)^n}$-relation

$$f(x)^q g(x) - f(x) g(x)^q = \left(f(x)^q g(x) - f(x) g(x)^q\right) \mod I(x). \tag{2}$$

By the factorisation of $X^q - X$ over $\mathbb{F}_q$, the LHS of Eq. (2) has irreducible factors of degree at most $D - d$. On the RHS one stipulates that it be zero mod $Q(x)$. This condition can be expressed as a bilinear quadratic system in the dk $\mathbb{F}_q$-components of the coefficients of f and the $(D - d)k$ $\mathbb{F}_q$-components of the coefficients of g. Since $Q(x)$ has D coefficients in $\mathbb{F}_{q^k}$ one expects there to be $O(1)$ solutions to this system when both f and g are monic. Hence by varying the leading coefficient of one of them, one expects many solutions.

The degree of the RHS of Eq. (2) depends on the representation of the field $\mathbb{F}_{(q^k)^n}$. Recall that in Joux's field representation, one has $h_0(X)$, $h_1(X)$ of very low degree δ_{h_0}, δ_{h_1} such that $h_1(X)X^q - h_0(X) \equiv 0 \pmod{I(X)}$, with $I(X)$ irreducible of degree n and $n \approx q$. Now on the RHS of Eq. (2) one replaces each occurrence of x^q by $h_0(x)/h_1(x)$, and thus the cofactor of $Q(x)$ on the RHS has degree $(D - d)(\max\{\delta_{h_0}, \delta_{h_1}\} - 1)$. For each solution to the bilinear quadratic system, it is tested for $(D - d)$-smoothness, and when it is, one has successfully represented $Q(x)$ as a product of at most q field elements of degree at most $D - d$ (ignoring the negligible number of factors from the cofactor).

Using our field representation, recall that $y = x^q$ and hence

$$f(x)^q = \sum_{i=0}^{d} f_i^q y^i \quad \text{and} \quad g(x)^q = \sum_{j=0}^{D-d} g_j^q y^j \,.$$

Then also using $x = p_1(y)$, the RHS of Eq. (2) becomes:

$$\left(\sum_{i=0}^{d} f_i^q y^i\right)\left(\sum_{j=0}^{D-d} g_j p_1(y)^j\right) - \left(\sum_{i=0}^{d} f_i p_1(y)^i\right)\left(\sum_{j=0}^{D-d} g_j^q y^j\right),$$

so that the cofactor of $Q(y)$ has degree $(D-d)(\delta_1 - 1)$ in y.

By repeating the above elimination technique recursively for each element occurring in the product until only degree one or degree two elements remain, the logarithm of $Q(x)$ is computed. So what is the optimal d? Joux's analysis [12] indicates that $d = O(q^{1/4}(\log q)^{1/2})$ should be used, giving an overall complexity of $\exp\left((c' + o(1))\, q^{1/4}(\log q)^{3/2}\right)$ for some c', which is $L_{q^{kq}}(1/4 + o(1), c')$, due to the presence of the extra $(\log q)^{1/2}$ factor, relative to Eq. (1).

However, one can instead set $d \approx \alpha_2 q^{1/4}$, as we now show (the constant α_2 is to be optimised later). Let $C(D,d)$ be the cost of expressing a degree D element as a product of elements of degree at most d, when the numerator f has degree d at each step. If $C_0(D,d)$ is the cost of resolving the corresponding bilinear quadratic system, we have

$$\begin{aligned} C(D,d) &= C_0(D,d) + q\, C(D-d,d) \\ &= C_0(D,d) + q\left(C_0(D-d,d) + q\, C(D-2d,d)\right) \\ &= \cdots = \sum_{i=0}^{\lfloor D/d \rfloor - 1} q^i C_0(D - id, d)\,. \end{aligned}$$

Since $C_0(D-id,d) \le C_0(D,d)$ for all i and since $\sum_{i=0}^{\lfloor D/d \rfloor - 1} q^i \le q^{D/d}$ we get the upper bound

$$C(D,d) \le q^{D/d} C_0(D,d)\,.$$

As in [12], we need the following essential lemma.

Lemma 1. ([25, Corollary 6.30]) *The arithmetic complexity (measured in $\mathbb{F}_q$-operations) of computing a Gröbner basis of a generic bilinear system $f_1, \ldots, f_{n_x+n_y} \in \mathbb{F}_q[x_0, \ldots, x_{n_x-1}, y_0, \ldots, y_{n_y-1}]$ with Faugere's F4 algorithm [4] is bounded by*

$$O\Big(\min(n_x, n_y)\,(n_x + n_y)\binom{n_x + n_y + \min(n_x, n_y) + 2}{\min(n_x, n_y) + 2}^{\omega}\Big),$$

where ω is the exponent of matrix multiplication.

Hence, using the estimate $\binom{a+2}{b+2} \le (\frac{a}{b})^2 \binom{a}{b} \le (\frac{a}{b})^2 (e\,\frac{a}{b})^b = e^b (\frac{a}{b})^{b+2}$, we have

$$C_0(D,d) = O\Big(k^2 Dd \binom{k(D+d)+2}{kd+2}^{\omega}\Big) = O\Big(k^2 Dd e^{k\omega d} \left(\frac{D+d}{d}\right)^{k\omega d + 2\omega}\Big)\,,$$

and, neglecting the lower order terms, we get

$$\log C_0(D,d) = \big(k\omega d \log(D/d)\big)(1+o(1))\,.$$

Therefore, we have

$$\begin{aligned}\log C(D,d) &= \big((D/d)\log q + k\omega d \log(D/d)\big)(1+o(1)) \\ &= \Big(\Big(\frac{\alpha_1}{\alpha_2} + \frac{k\omega\alpha_2}{4}\Big) q^{1/4} \log q\Big)(1+o(1))\,,\end{aligned}$$

and in particular, for the optimal choice $\alpha_2 = (4\alpha_1/k\omega)^{1/2}$, we get

$$\log C(D,d) = \big((k\omega\alpha_1)^{1/2} q^{1/4} \log q\big)(1+o(1))\,.$$

Thus, taking into account Eq. (1), we arrive at the complexity

$$C(D,d) = L_{q^{kq}}(1/4\,,\, k^{1/4}(\omega\alpha_1)^{1/2})\,. \tag{3}$$

Observe that the number of degree $d \approx \alpha_2 q^{1/4}$ elements in such an expression for the initial degree $D \approx \alpha_1 q^{1/2}$ element is $O(q^{(\alpha_1/\alpha_2)q^{1/4}})$. Note that this choice of d represents the optimal balance between the number of nodes in the descent tree at level d and the cost of resolving the bilinear systems.

Moreover, exactly the same argument shows that $C(\alpha_j q^{1/2^j}, \alpha_{j+1} q^{1/2^{j+1}}) = L_{q^{kq}}(1/2^{j+1})$, and so the cost of expressing each of the $L_{q^{kq}}(1/4)$ degree $\alpha_2 q^{1/4}$ elements in terms of elements of degree $\alpha_3 q^{1/8}$ is $L_{q^{kq}}(1/8)$, and therefore for any $j > 1$ the total cost down to degree $\alpha_j q^{1/2^j}$ never exceeds $L_{q^{kq}}(1/4)$. After $j = \lceil \log_2 \log_2 q \rceil$ of the above sequence of steps we have $\lfloor q^{1/2^j} \rfloor = 1$, and the total cost is precisely that given in Eq. (3).

As the complexity of the initial splitting of a target element into a product of elements of degree at most $\alpha_0 q^{3/4}$ is $L_{q^{kq}}(1/4)$, as is the complexity of classical descent from degree $\alpha_0 q^{3/4}$ to degree $\alpha_1 q^{1/2}$, the above tighter analysis demonstrates that for the fields considered, Joux's algorithm has complexity $L_{q^{kq}}(1/4)$ as well, for both his and our field representations. We have omitted the determination of the optimal parameters α_0 and α_1, since this is beyond our focus on proving that the full algorithm is $L(1/4)$.

References

1. Barreto, P.S.L.M., Galbraith, S.D., Ó' hÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Des. Codes Cryptogr. **42**(3), 239–271 (2007)

2. Bluher, A.W.: On $x^{q+1} + ax + b$. Finite Fields Appl. **10**(3), 285–305 (2004)
3. Bosma, W., Cannon, J., Playoust, C.: The magma algebra system. I. The user language. J. Symbolic Comput. **24**(3–4), 235–265 (1997)
4. Faugére, J.C.: A new efficient algorithm for computing Gröbner bases (F_4). J. Pure Appl. Algebra **139**(1–3), 61–88 (1999)
5. Gaudry, P., Hess, F., Smart, N.P.: Constructive and destructive facets of weil descent on elliptic curves. J. Cryptol. **15**(1), 19–46 (2002)
6. Göloğlu, F., Granger, R., McGuire, G., Zumbrägel, J.: On the function field sieve and the impact of higher splitting probabilities: application to discrete logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 109–128. Springer, Heidelberg (2013)
7. Göloğlu, F., Granger, R., McGuire, G., Zumbrägel, J.: Discrete Logarithms in $GF(2^{1971})$. NMBRTHRY list, 19 Feb 2013
8. Göloğlu, F., Granger, R., McGuire, G., Zumbrägel, J.: Discrete Logarithms in $GF(2^{6120})$. NMBRTHRY list, 11 Apr 2013
9. Granlund, T.: The GMP development team: GNU MP: The GNU Multiple Precision Arithmetic Library, 5.0.5 edn. http://gmplib.org/ (2012)
10. Helleseth, T., Kholosha, A.: $x^{2^l+1} + x + a$ and related affine polynomials over (2^k). Cryptogr. Commun. **2**(1), 85–109 (2010)
11. 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)
12. Joux, A.: A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. Cryptology ePrint Archive, report 2013/095. http://eprint.iacr.org/ (2013)
13. Joux, A.: Personal communication (2013)
14. Joux, A.: Discrete Logarithms in $GF(2^{1778})$. NMBRTHRY list, 11 Feb 2013
15. Joux, A.: Discrete Logarithms in $GF(2^{4080})$. NMBRTHRY list, 22 Mar 2013
16. Joux, A.: Discrete Logarithms in $GF(2^{6168})$. NMBRTHRY list, 21 May 2013
17. 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)
18. LaMacchia, B.A., Odlyzko, A.M.: Solving large sparse linear systems over finite fields. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 109–133. Springer, Heidelberg (1991)
19. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Nat. Bur. Stan. **45**, 255–282 (1950)
20. Lenstra, A.K., Lenstra Jr, H.W. (eds.): The Development of the Number Field Sieve. LNM, vol. 1554. Springer, Heidelberg (1993)
21. Lenstra Jr, H.W.: Factoring integers with elliptic curves. Ann. Math. (2) **126**(3), 649–673 (1987)
22. Lenstra Jr, H.W.: Finding isomorphisms between finite fields. Math. Comp. **56**(193), 329–347 (1991)
23. Popovyan, I.: Efficient parallelization of lanczos type algorithms. Cryptology ePrint Archive, Report 2011/416. http://eprint.iacr.org/ (2011)
24. Shoup, V.: NTL: A library for doing number theory, 5.5.2 edn. http://www.shoup.net/ntl/ (2009)
25. Spaenlehauer, P.J.: Solving multihomogeneous and determinantal systems algorithms - complexity - applications. Ph.D. thesis, Université Pierre et Marie Curie (UPMC) (2012)