

# Computation of a 30 750-Bit Binary Field Discrete Logarithm

Robert Granger, Thorsten Kleinjung, Arjen K Lenstra, Benjamin Wesolowski, Jens Zumbrägel

▶ **To cite this version:**

Robert Granger, Thorsten Kleinjung, Arjen K Lenstra, Benjamin Wesolowski, Jens Zumbrägel. Computation of a 30 750-Bit Binary Field Discrete Logarithm. Mathematics of Computation, 2021, 90 (332), 10.1090/mcom/3669 . hal-02945361

**HAL Id: hal-02945361**
**https://hal.science/hal-02945361**

Submitted on 22 Sep 2020

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Computation of a 30 750-Bit Binary Field Discrete Logarithm

Robert Granger[1], Thorsten Kleinjung[2], Arjen K. Lenstra[2],
Benjamin Wesolowski[3], and Jens Zumbrägel[4]

[1] Surrey Centre for Cyber Security
Department of Computer Science, University of Surrey, United Kingdom
r.granger@surrey.ac.uk
[2] Laboratory for Cryptologic Algorithms,
School of Computer and Communication Sciences, EPFL, Switzerland
thorsten.kleinjung@epfl.ch
[3] Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, F-33400, Talence, France
INRIA, IMB, UMR 5251, F-33400, Talence, France
benjamin.wesolowski@math.u-bordeaux.fr
[4] Faculty of Computer Science and Mathematics, University of Passau, Germany
jens.zumbraegel@uni-passau.de

**Abstract.** This paper reports on the computation of a discrete logarithm in the finite field $\mathbb{F}_{2^{30750}}$, breaking by a large margin the previous record, which was set in January 2014 by a computation in $\mathbb{F}_{2^{9234}}$. The present computation made essential use of the elimination step of the quasi-polynomial algorithm due to Granger, Kleinjung and Zumbrägel, and is the first large-scale experiment to truly test and successfully demonstrate its potential when applied recursively, which is when it leads to the stated complexity. It required the equivalent of about 2900 core years on a single core of an Intel Xeon Ivy Bridge processor running at 2.6 GHz, which is comparable to the approximately 3100 core years expended for the discrete logarithm record for prime fields, set in a field of bit-length 795, and demonstrates just how much easier the problem is for this level of computational effort. In order to make the computation feasible we introduced several innovative techniques for the elimination of small degree irreducible elements, which meant that we avoided performing any costly Gröbner basis computations, in contrast to all previous records since early 2013. While such computations are crucial to the $L(\frac{1}{4} + o(1))$ complexity algorithms, they were simply too slow for our purposes. Finally, this computation should serve as a serious deterrent to cryptographers who are still proposing to rely on the discrete logarithm security of such finite fields in applications, despite the existence of two quasi-polynomial algorithms and the prospect of even faster algorithms being developed.

**Keywords:** Discrete logarithm problem, finite fields, binary fields, quasi-polynomial algorithm

## 1 Introduction

Let $\mathbb{F}_2$ denote the finite field consisting of two elements, let $\mathbb{F}_{2^{30}} := \mathbb{F}_2[T]/(T^{30} + T + 1)$ and let $t$ be a root of the irreducible polynomial defining the extension. Furthermore, let $\mathbb{F}_{2^{30750}} := \mathbb{F}_{2^{30}}[X]/(X^{1025}+X+t^3)$ and let $x$ be a root of the irreducible polynomial defining this extension, and consider the presumed generator $g := x+t^9$ of the multiplicative group of $\mathbb{F}_{2^{30750}}$. To select a target element that cannot be 'cooked up' we follow the traditional approach of using the digits of the mathematical constant $\pi$, which in the present case leads to

$$h_\pi := \sum_{i=0}^{30749} \left(\lfloor \pi \cdot 2^{i+1} \rfloor \bmod 2\right) \cdot t^{29-(i \bmod 30)} \cdot x^{\lfloor i/30 \rfloor} \in \mathbb{F}_{2^{30750}}.$$

On 28th May 2019 we completed the computation of the discrete logarithm of $h_\pi$ with respect to the base $g$, i.e., an integer solution $\ell$ to $h_\pi = g^\ell$. The smallest non-negative solution is given in Appendix B, along with a verification script for the computational algebra system Magma [3]. The total running time for this computation was the equivalent of about 2900 core years on a single core of an Intel Xeon Ivy Bridge or Haswell processor running at 2.6 GHz or 2.5 GHz. For the sake of comparison, the previous record, which was set in $\mathbb{F}_{2^{9234}}$ in January 2014, required about 45 core years [19]. Other large scale computations of this type include: the

factorisation of a 768-bit RSA modulus, which took about 1700 core years and was completed in 2009 [26]; the factorisation of 17 Mersenne numbers with bit-lengths between 1007 and 1199 using Coppersmith's 'factorisation factory' idea [8], which took about 7500 core years and was completed in early 2015 [27]; the computation of a discrete logarithm in a 768-bit prime field, which took about 5300 core years and was completed in 2016 [28]; the computation of a discrete logarithm in a 795-bit prime field, which took about 3100 core years, and the factorisation of a 795-bit RSA modulus, which took about 900 core years, both completed in December 2019 [4]; and finally, the factorisation of an 829-bit RSA modulus, which took about 2700 core years and was completed in February 2020 [5].

A natural question that the reader may have is why did we undertake such a large-scale computation? The answer to this question is threefold. Firstly, between 1984 and 2013, the fastest algorithm for solving the discrete logarithm problem (DLP) in characteristic two (and more generally in fixed characteristic) fields was due to Coppersmith [7]. It has heuristic complexity $L_Q(\frac{1}{3})$, where for $\alpha \in [0,1]$ and $Q$ the cardinality of the field, this notation is defined by

$$L_Q(\alpha) := \exp\left(O((\log Q)^{\alpha}(\log\log Q)^{1-\alpha})\right)$$

as $Q \to \infty$; this is the usual complexity measure that interpolates between polynomial time (for $\alpha = 0$) and exponential time (for $\alpha = 1$), and is said to be subexponential when $0 < \alpha < 1$. In 2013 and 2014 a series of breakthroughs occurred, indicating that the DLP in fixed characteristic fields was considerably easier to solve than had previously been believed [10,20,11,1,15,17,24,16]. These techniques were demonstrated with the setting of several world records for computations of this type [21,13,23,12,22,18,19], using algorithms that had heuristic complexity at most $L_Q(\frac{1}{4} + o(1))$, where $o(1) \to 0$ as $Q \to \infty$. Amongst all of the breakthroughs from this period, the standout results were two independent and distinct quasi-polynomial algorithms for the DLP in fixed characteristic: the first due to Barbulescu, Gaudry, Joux and Thomé (BGJT) [1]; and the second due to Granger, Kleinjung and Zumbrägel (GKZ) [17,16]. Both algorithms have heuristic complexity $L_Q(o(1))$, although the latter is rigorous once an appropriate field representation is known. As further discussed below, such representations exist, for instance, for Kummer extensions. While the computational records demonstrated the practicality of the $L(\frac{1}{4} + o(1))$ techniques, it was the promise of the quasi-polynomial algorithms that effectively killed off the use of small characteristic fields in discrete logarithm-based cryptography, even though these algorithms had not been seriously tested. In particular, the descent step, or 'building block' of the BGJT method has to our knowledge never been used for the individual logarithm stage. The descent step of the GKZ method, on the other hand, has been used during the individual logarithm stage for some small degree eliminations for relatively small fields [25,24], but not with a view to properly testing its applicability recursively, as it is intended to be used. Since the GKZ algorithm becomes practical for far smaller bit-lengths than the BGJT algorithm, one aspect of our motivation was to test its practicality by seeing in how large a field we could solve discrete logarithms, within a reasonable time. The act of doing so allows one to assess the practical impact of the algorithm and indeed, its success in this case demonstrates that it can be applied effectively at a large scale. It also demonstrates just how much easier the DLP is in binary fields than for prime fields, for this level of computational effort. We note that a recent paper of the second and fourth listed authors proves that the complexity of the DLP in fixed characteristic fields is at most quasi-polynomial, without restriction on the form of the extension degree – in contrast to the GKZ algorithm – thus complementing the practical impact of the present computation with a rigorous theoretical algorithm for the general case [29].

Secondly, we wanted to set a significant new record because in our experience of attempting such computations, one almost always encounters many unexpected obstacles that necessitate having new insights and developing new techniques in order to overcome them, both of which enrich our knowledge and understanding, as well as the state of the art. Indeed, it was the solving of a DLP in $\mathbb{F}_{2^{4404}}$ in 2014 [15] that led to the discovery of the GKZ algorithm. As Knuth has

stated, *"The best theory is inspired by practice. The best practice is inspired by theory."*, which is as true in this field as it is in any other. In this work, we developed several innovative techniques for the elimination of small degree irreducible polynomials, which were absolutely vital to the feasibility of computing a discrete logarithm within such an enormous field. In particular, our *new insights and contributions* include the following.

- A simple theoretical and efficient algorithmic characterisation of all so-called Bluher values.
- A highly efficient direct method for eliminating degree 2 polynomials over $\mathbb{F}_{q^3}$ on the fly, which applies with probability $\approx \frac{1}{2}$.
- A highly efficient backup method for eliminating degree 2 polynomials over $\mathbb{F}_{q^3}$ when the direct method fails, making the bottleneck of the computation feasible. In addition we found an interesting theoretical explanation for the observed probabilities.
- A fast probabilistic elimination of degree 2 polynomials over $\mathbb{F}_{q^k}$ for $k > 3$.
- A novel use of interpolation to efficiently compute roots of the special polynomials that arise.
- An extremely efficient and novel degree 3 elimination method, thus mollifying the new bottleneck in the computation, which was also applied to polynomials of degree 6, 9 and 12.
- A new technique for eliminating polynomials of degrees 5 and 7, which together with the techniques for the other small degree polynomials meant that *no costly Gröbner basis computations were performed* at all. Although such computations are essential to the $L_Q(\frac{1}{4} + o(1))$ complexity algorithms, they were simply too slow for our purposes.
- A highly optimised classical descent using the above elimination costs as input in order to apply a dynamic programming approach.

Finally, although discrete logarithm-based cryptography in small characteristic finite fields has effectively been rendered unusable, there are constructive applications in cryptography for fast discrete logarithm algorithms. For example, knowledge of certain discrete logarithms in binary fields can be exploited to create highly efficient, constant time masking functions for tweakable blockciphers [14]. Further afield, there are several applications which would benefit from efficient discrete logarithm algorithms, although not necessarily in the fixed characteristic case. For example, in matrix groups a problem in $\mathrm{GL}(n, q)$ can often be mapped to $\mathrm{GL}(1, q^n)$ and naturally becomes a DLP in $\mathbb{F}_{q^n}$. Also, discrete logarithms are needed in various scenarios when reducing the unit group of an order of a number field modulo a prime. In finite geometry one encounters near-fields, where multiplication usually requires computing discrete logarithms, and there are other applications in representation theory, group theory and Lie algebras in the modular case. The relevance of the DLP to computational mathematics therefore extends far beyond cryptography. Despite the existence of the quasi-polynomial algorithms, the use of prime degree extensions of $\mathbb{F}_2$ has recently been proposed in the design of a secure compressed encryption scheme of Canteaut *et al.* [6, Sec. 5]. For 80 bits of security a prime extension of degree $\approx 16\,000$ was suggested (and of degree $4\,000\,000$ for 128-bit security), very conservatively based on the polynomial time first stage of index calculus as described in [24], but ignoring the dominating quasi-polynomial individual logarithm stage. Even though the security of such fields is massively underestimated and the resulting schemes are prohibitively slow, it is interesting, to say the least, that these proposals were made. Although the computation reported here is not for a prime degree extension of $\mathbb{F}_2$, our result should nevertheless be regarded as a serious deterrent against such applications. Indeed, the central remaining open problem in this area, which is to find a polynomial time algorithm for the DLP in fixed characteristic fields (either rigorous or heuristic) is far more likely to be solved now than it was before 2013.

Another natural question that the reader may have is why 30 750, exactly? In contrast to the selection of $h_\pi$, this extension degree was most certainly 'cooked up' specifically for setting a new record[5]. In particular, $30\,750 = 3 \cdot 10 \cdot (2^{10} + 1)$ so that the field is of the form $\mathbb{F}_{q^{3(q+1)}}$, thus

[5] The fact that 30 750 is precisely the seating capacity of the first listed author's home football stadium is purely a coincidence; see `https://en.wikipedia.org/wiki/Falmer_Stadium`.

permitting the use of a twisted Kummer extension, in which it is much easier to solve logarithms than in other extensions of the same order of magnitude. Firstly, as explained in §5 the factor base which consists of all monic degree one elements possesses an automorphism of order 3075. This reduces the cardinality of the factor base from $2^{30}$ to a far more manageable 349 185. Secondly, such field representations ensure that when eliminating an element during the descent step, the cofactors have minimal degree and therefore will be smooth with maximum probability (under a uniformity assumption when not using the GKZ elimination step). Thirdly, there are no so-called 'traps' during the descent, i.e., elements which can not be eliminated [16]. The first two reasons explain why Kummer and twisted Kummer extensions have been used for all of the records since 2013. Observe that for the previous record we have $9234 = 2 \cdot 9 \cdot (2^9 + 1)$. However, whereas for the 9234-bit computation the factor base consisted of all degree one and irreducible degree two elements, for the present computation it would be too costly to compute and store the logarithms of such a factor base, even when incorporating the automorphism. So it was essential that the logarithms of degree two elements could be computed 'on the fly', i.e., as and when needed, without batching. For a base field of the form $\mathbb{F}_{q^3}$ this is non-trivial as each degree two element only has a half chance of being eliminable when the method of [10] is applied. For our target field, a backup approach similar in spirit to that used for the $6120 = 3 \cdot 8 \cdot (2^8 - 1)$-bit record can be used [12,11], and the chosen parameters allow for this with an acceptable probability. However, in order to make the computation feasible, as already mentioned it was essential to develop new techniques for the elimination of degree two and other small degree polynomials and to optimise them algorithmically, arithmetically and implementation-wise. For the number of core years used we therefore believe with high confidence that we have solved as large a DLP as is possible with current state of the art techniques.

The sequel is organised as follows. In §2 we present some results on so-called Bluher polynomials that are essential to the theory and practice of our computation. In §3 we recall the GKZ algorithm, explaining the importance of degree two elimination and how this leads to a quasi-polynomial algorithm. We describe the field setup in §4, and in §5 describe how we computed the logarithms of the factor base elements, namely the degree one elements. Subsequently we present our degree two elimination methods in §6 and more generally even degree element elimination in §7, and in §8 detail how we eliminated small odd degree elements. In §9 we explain how we efficiently compute the roots of Bluher polynomials in various scenarios, while in §10 we detail the classical descent strategy, analysis and timings. In §11 we briefly discuss the security of the proposal of Canteaut *et al.*, and finally in §12 we make some concluding remarks.

## 2 Bluher Polynomials and Values

Let $q$ be a prime power, let $k \geq 3$ and consider the polynomial $F_B(X) \in \mathbb{F}_{q^k}[X]$ defined by

$$F_B(X) := X^{q+1} - BX + B. \tag{1}$$

We call such a polynomial a Bluher polynomial. A basic question that arises repeatedly for us is for what values $B \in \mathbb{F}_{q^k}^{\times}$ does $F_B(X)$ split over $\mathbb{F}_{q^k}$, i.e., factor into a product of $q+1$ linear polynomials in $\mathbb{F}_{q^k}[X]$? With this in mind we define $\mathcal{B}_k$ to be the set of all $B \in \mathbb{F}_{q^k}^{\times}$ such that $F_B(X)$ splits over $\mathbb{F}_{q^k}$, and call the members of $\mathcal{B}_k$ Bluher values. It turns out that there are some simple characterisations of $\mathcal{B}_k$ which enable various computations to be executed very efficiently. We first recall a result of Bluher.

**Theorem 1.** *[2] 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*

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

One characterisation of $\mathcal{B}_k$ is the following generalisation of a theorem due to Helleseth and Kholosha.

**Theorem 2.** *[16, Lemma. 4.1] We have*

$$\mathcal{B}_k = \left\{ \frac{(u-u^{q^2})^{q+1}}{(u-u^q)^{q^2+1}} \,\middle|\, u \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^2} \right\}.$$

Theorem 2 is useful for sampling from $\mathcal{B}_k$. However, sometimes it is desirable to test whether a given element of $\mathbb{F}_{q^k}$ is a Bluher value, without factorising (1) and without first enumerating $\mathcal{B}_k$ and checking membership, which may be intractable when $k$ is so large that there are prohibitively many Bluher values to precompute. To this end, we define the following polynomials. Let $P_1(X) = 1$, $P_2(X) = 1$, and for $i \geq 3$ define $P_i(X)$ by the recurrence

$$P_i(X) = P_{i-1}(X) - X^{q^{i-3}} P_{i-2}(X). \tag{2}$$

**Theorem 3.** *An element $B \in \mathbb{F}_{q^k}^\times$ is a Bluher value if and only if $P_k(\frac{1}{B}) = 0$.*

*Proof.* A simple induction shows that the degree of $P_k$ equals the number of Bluher values given in Theorem 1, so it suffices to prove the only if part. Let $B$ be a Bluher value and $C = \frac{1}{B}$. Using $X^q \equiv \frac{X-1}{CX} \pmod{F_B}$ and induction, for $i \geq 2$ one obtains

$$X^{q^i} \equiv \frac{P_{i+1}(C)X - P_i^q(C)}{C^{q^{i-1}}(P_i(C)X - P_{i-1}^q(C))} \pmod{F_B}.$$

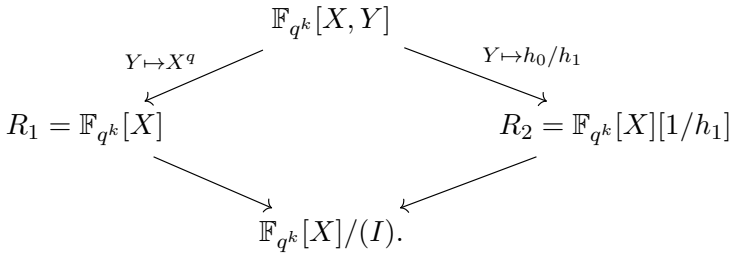
Since $X \equiv X^{q^k} \pmod{F_B}$ by assumption, $C^{q^{k-1}}(P_k(C)X - P_{k-1}^q(C))X \equiv P_{k+1}(C)X - P_k^q(C) \pmod{F_B}$ and thus $P_k^q(C) = 0$.

One can efficiently test if an element $B \in \mathbb{F}_{q^k}^\times$ is a Bluher value simply by evaluating $P_k(\frac{1}{B})$ using the recurrence (2). By rewriting the recurrence in a matrix form it is possible to apply fast methods for evaluating it. However, since $k$ is quite small for most of the computations, we did not use this. Furthermore, one can obviously compute $\mathcal{B}_k$ by applying the recurrence (2) and factorising $P_k$.

## 3 The GKZ Algorithm

In this section we sketch the main idea behind the GKZ algorithm, referring the reader to the original paper for rigorous statements and proofs [16].

Assume there exist coprime $h_0, h_1 \in \mathbb{F}_{q^k}[X]$ of degree at most two such that there exists a monic irreducible polynomial $I$ of degree $n$ with $I \mid h_1 X^q - h_0$. Let $\mathbb{F}_{q^{kn}} := \mathbb{F}_{q^k}[X]/(I) = \mathbb{F}_{q^k}(x)$ where $x$ is a root of $I$. Then we have the following commutative diagram:




Assume further for the moment that for any $d \geq 1$, it is possible in time polynomial in $q$ and $d$ to express a given irreducible degree two element $Q \in \mathbb{F}_{q^{kd}}[X]$ as a product of at most $q+2$ linear elements in $\mathbb{F}_{q^{kd}}[X]$ mod $I$, i.e.,

$$Q \equiv \prod_{i=1}^{q+2} (X + a_i) \pmod{I}, \quad \text{with } a_i \in \mathbb{F}_{q^{kd}}. \tag{3}$$

We have the following.

**Proposition 1.** *[16, Prop. 3.2] For the stated field representation, let $d \geq 1$ and let $Q \in \mathbb{F}_{q^k}[X]$ be an irreducible polynomial of degree $2d$. Then $Q$ can be rewritten in terms of at most $q+2$ irreducible polynomials of degrees dividing $d$ in an expected running time polynomial in $q$ and in $d$.*

We now sketch the proof. Over $\mathbb{F}_{q^{kd}}$, we have the factorisation $Q = \prod_{i=1}^{d} Q_i$, where $Q_i \in \mathbb{F}_{q^{kd}}[X]$ are irreducible quadratics. Applying the above rewriting assumption to any one of the elements $Q_i$ gives an instance of the product in (3), with $Q_i$ on the l.h.s. Taking the norm map, i.e., the product of all conjugates of each term under $\mathrm{Gal}(\mathbb{F}_{q^{kd}}/\mathbb{F}_{q^k})$, produces the original $Q$ on the l.h.s. and for each term on the r.h.s., a $d_1$-th power of an irreducible polynomial in $\mathbb{F}_{q^k}[X]$ of degree $d_2$, where $d_1 d_2 = d$, which is what the proposition states.

Now, if one begins with an irreducible polynomial $Q$ of degree $2^e$ with $e \geq 1$, then recursively applying Proposition 1 allows one to express $Q$ as a product of at most $(q+2)^e$ linear polynomials. Moreover, given $g \in \mathbb{F}_{q^{kn}}^{\times}$ and $h \in \langle g \rangle$ an element whose logarithm with respect to $g$ is to be computed, one can efficiently find an irreducible representative of $h \bmod I$ of degree $2^e$, provided that $2^e > 4n$, by adding random multiples of $I$ to $h$ so that the degree of $h+rI$ is $2^e$ [16, Lem. 3.3].

The logarithms of the linear elements – the factor base – remain to be computed, but this can be accomplished in polynomial time in various ways [10,20], or can be obviated altogether by repeating the descent to linear elements as many times as the cardinality of the factor base [9], but the latter is usually only of theoretical interest. Note that in practice the above descent method is not optimal as for high degree element elimination the classical methods are more efficient, since they produce far fewer descendants.

The crux of this approach, namely eliminating a degree two element as expressed in (3), can be achieved in various ways originating with [10], as follows. Let $a, b, c \in \mathbb{F}_{q^k}$ and consider the polynomial $X^{q+1} + aX^q + bX + c$. Firstly, since $X^q \equiv h_0/h_1 \pmod{I}$ we have

$$X^{q+1} + aX^q + bX + c \equiv \tfrac{1}{h_1}\big((X+a)h_0 + (bX+c)h_1\big) \pmod{I}, \tag{4}$$

with the numerator of the r.h.s. of (4) of degree at most three. We would like to impose that this numerator is divisible by $Q$. Therefore, let $L_Q \in \mathbb{F}_{q^k}[X]^2$ be the lattice defined by

$$L_Q := \{(w_0, w_1) \in \mathbb{F}_{q^k}[X]^2 \mid w_0 h_0 + w_1 h_1 \equiv 0 \pmod{Q}\}.$$

In general, $L_Q$ has a basis of the form $(1, u_0X + u_1), (X, v_0X + v_1)$ with $u_i, v_i \in \mathbb{F}_{q^k}$ and thus in order for the r.h.s. of (4) to be divisible by $Q$ we must choose $a, b, c$ such that

$$(X + a, bX + c) = a(1, u_0X + u_1) + (X, v_0X + v_1). \tag{5}$$

When this is so, the cofactor of $Q$ has degree at most one.

Secondly, provided that $b \neq a^q$ and $c \neq ab$, the polynomial $X^{q+1} + aX^q + bX + c$ may be transformed by using the substitution

$$X \longmapsto \frac{ab - c}{b - a^q} X - a \tag{6}$$

into a scalar multiple of the Bluher polynomial $X^{q+1} - BX + B$, where

$$B := \frac{(b - a^q)^{q+1}}{(c - ab)^q}. \tag{7}$$

Thus, if these two conditions on $a, b, c$ hold then $X^{q+1} + aX^q + bX + c$ splits whenever $B \in \mathcal{B}_k$. Combining (5) and (7) yields the condition

$$B = \frac{(-a^q + u_0 a + v_0)^{q+1}}{(-u_0 a^2 + (u_1 - v_0)a + v_1)^q}. \tag{8}$$

In order to eliminate $Q$, one chooses $B \in \mathcal{B}_k$ to obtain a univariate polynomial in $a$, which can be checked for roots in $\mathbb{F}_{q^k}$ in various ways, for instance by computing the GCD with $a^{q^k} - a$. Note that the more Bluher values there are, the higher the chance is of eliminating $Q$. By Theorem 1 the hardest case is for $k = 3$ as there is only one such Bluher value, namely 1, which is the case we address in this paper, using a backup elimination method à la [11]. By this rationale, in general when Proposition 1 is applied, the elimination of degree two $Q_i$ over $\mathbb{F}_{q^{kd}}$ becomes 'easier' as $d$ grows, although the computational cost per Bluher value increases as the base field increases in size.

## 4 Field Setup and Factors

Recall that the group in which we look at the discrete logarithm problem is the (cyclic) unit group of the finite field $\mathbb{F}_{2^{30750}}$, thus its group order is $N := 2^{30750} - 1$. We used the factorisation algorithm of Magma, which is assisted by the 'Cunningham tables' for Mersenne numbers, to obtain the 58 known prime factors of $N$ up to bit-length 135 (see Appendix A).[6]

Also recall that to represent the field $\mathbb{F}_{2^{30750}}$, we first let $\mathbb{F}_{2^{30}} := \mathbb{F}_2[t] = \mathbb{F}_2[T]/(T^{30} + T + 1)$ where $t := [T] \in \mathbb{F}_{2^{30}}$, and then, setting $\gamma := t^3$, we define the target field as the extension

$$\mathbb{F}_{2^{30750}} := \mathbb{F}_{2^{30}}[x] = \mathbb{F}_{2^{30}}[X]/(X^{1025} + X + \gamma)$$

where $x := [X] \in \mathbb{F}_{2^{30750}}$.[7] The field element $g := x + t^9$ is then a supposed generator for the group, as $g^{N/p} \neq 1$ for the 58 listed prime factors $p \mid N$.

The schedule and the running times of our computation of the discrete logarithm of the target element $h_\pi \in \mathbb{F}_{2^{30750}}$ from the introduction are summarised in Table 1. In the subsequent sections we present the corresponding steps in more detail.

**Table 1.** Overall schedule and running times.

| Step | when | core hours |
|---|---|---|
| Relation Generation | 23 Feb 2016 | 1 |
| Linear Algebra | 27 Feb 2016 - 24 Apr 2016 | 32 498 |
| Initial Split | 09 Sep 2016 - 20 Sep 2016 | 248 140 |
| Classical Descent | 13 Oct 2016 - 23 Jan 2019 | 17 077 836 |
| Small Degree Descent | 27 Jan 2019 - 28 May 2019 | 8 122 744 |
| *total running time* | | 25 481 219 |

## 5 Logarithms of Degree 1 Elements

As usual this consists of relation generation followed by a linear algebra elimination.

### 5.1 Relation Generation

The relation generation step is based on the following. As before, let $\mathbb{F}_q$ be a finite field, let $k \geq 3$ and let $a, b, c \in \mathbb{F}_{q^k}$. Then over $\mathbb{F}_{q^k}$, the polynomial $X^{q+1} + aX^q + bX + c$ splits if and only if its affine transformation $X^{q+1} - BX + B$ splits, with $B$ as given in (7). According to

[6] We remark that this partial factorisation is sufficient, since the expected running time of a generic discrete logarithm algorithm on a subgroup of order the largest listed prime already exceeds our total running time.

[7] Note that this convenient field representation for the discrete logarithm problem can be assumed without loss of generality, as it is possible to efficiently map between arbitrary field representations given by irreducible polynomials, cf. [32].

Theorem 1, for random $a, b, c$ this occurs with probability about $q^{-3}$, in which case the set of roots of $X^{q+1} + aX^q + bX + c$ is, by the transformation (6), given by

$$\{\lambda z - a \mid z \in \mathbb{F}_{q^k} \text{ a root of } X^{q+1} - BX + B\}, \quad \text{where} \quad \lambda := \frac{ab - c}{b - a^q}.$$

In our case we have $\mathbb{F}_{q^k} = \mathbb{F}_{2^{30}}$ and regard all linear polynomials $x+u$, for $u \in \mathbb{F}_{2^{30}}$, as the factor base. Since in the target field one has

$$x^{1024} = \frac{x + \gamma}{x},$$

one could use $q = 2^{10}$ and $k = 3$ for a fast relation generation. However, in order to decrease the row-weight of the resulting matrix, we prefer to choose $\tilde{k} = 6$ and $\tilde{q} = 2^5$, so that $x^{\tilde{q}^2} = \frac{x+\gamma}{x}$. The relation generation then makes use of the identity

$$(x^{\tilde{q}+1} + ax^{\tilde{q}} + bx + c)^{\tilde{q}} = \tfrac{1}{x}\big((b^{\tilde{q}} + 1)x^{\tilde{q}+1} + \gamma x^{\tilde{q}} + (a^{\tilde{q}} + c^{\tilde{q}})x + a^{\tilde{q}}\gamma\big),$$

which provides a relation of weight 66 each time both sides split.

Letting again $q = 2^{10}$, the Galois group $\mathrm{Gal}(\mathbb{F}_{2^{30750}}/\mathbb{F}_q)$ of order 3075 acts on the factor base elements. Indeed, we have

$$(x + u)^q = \tfrac{u^q+1}{x}\big(x + \tfrac{\gamma}{u^q+1}\big),$$

by which we can express all logarithms in a Galois orbit, up to a small cofactor, in terms of the logarithms of a single representative (and of $x$). This way we only need to consider the 349 185 orbits as variables. Within a running time of 63 minutes on an ordinary desktop computer the relation generation was finished.

### 5.2 Linear Algebra

The relation generation produced a $349\,195 \times 349\,184$ matrix with 66 nonzero entries in each row, each being of the form $\pm 2^e \in \mathbb{Z}/N\mathbb{Z}$ with some $e \in \mathbb{Z}/30750\mathbb{Z}$. We solved the linear algebra system for this matrix using the Lanczos method [31,30]. More precisely, with the intention to speed up modular computations, we used the factorisation

$$N = N_1 \cdot N_2 \cdot N_3 := (2^{10250} - 1) \cdot (2^{10250} + 2^{5025} + 1) \cdot (2^{10250} - 2^{5025} + 1)$$

and computed a solution vector for each of three moduli $M_i \mid N_i$ having small cofactors. This took 30 393 core hours on single computing nodes having 16 or 24 cores.

Then by the Chinese Remainder Theorem we combined the resulting solutions with the logarithms modulo the 38 prime factors of $N$ up to bit-length 43, listed in Appendix A. For the 18 larger ones of these moduli – the smallest being 2 252 951 – we computed the logarithms using the Lanczos algorithm as well. For the remaining 20 small factors, i.e., up to 165 313, we simply generated the corresponding subgroups in the target field and read off the discrete logarithms from the table. This way, and making use of the Galois orbits, we obtained the logarithms $\log(x + u)$ for all factor base elements, where $u \in \mathbb{F}_{2^{30}}$. The remaining computations accounted for a total of 2105 core hours on a single 16-core computing node.

## 6 Degree 2 Elimination Methods

As it is the bottleneck, in order to have a feasible computation it is indispensable to have an extremely efficient degree 2 elimination algorithm and implementation. In this section we apply the sketch of degree 2 elimination given in §3 to our target field and describe algorithms which are more efficient than those presented in [11,15]. Most of the techniques are more general and are presented in the setting of an arbitrary finite field $\mathbb{F}_{q^k}$ and $x^{q+1} - x - \gamma = 0$. Let $Q = X^2 + q_1X + q_0 \in \mathbb{F}_{q^k}[X]$ be an arbitrary irreducible quadratic polynomial to be eliminated, i.e., to be written as a product of linear elements.

### 6.1 Direct Method

Recall that $x^q = \frac{x+\gamma}{x}$, so that

$$x^{q+1} + ax^q + bx + c = \tfrac{1}{x}\big((b+1)x^2 + (a+c+\gamma)x + \gamma a\big) = \frac{b+1}{x}\left(x^2 + \frac{a+c+\gamma}{b+1}x + \frac{\gamma a}{b+1}\right). \tag{9}$$

Since the denominator $x$ on the r.h.s. of (9) is in the factor base, we know its logarithm. There is also no lattice to consider as the quadratic term must be $Q$, leading to the equations $q_1 = \frac{a+c+\gamma}{b+1}$ and $q_0 = \frac{\gamma a}{b+1}$. In order for the l.h.s. of (9) to split over $\mathbb{F}_{q^3}$, since there is only one Bluher value we must have $(b - a^q)^{q+1} = (c - ab)^q$. Writing $b = \frac{1}{q_0}(\gamma a - q_0)$ and $c = \frac{1}{q_0}(-q_0 a - \gamma q_0 + \gamma q_1 a)$ leads to a univariate polynomial equation in $a$, namely

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

This equation may be solved by computing the GCD with $a^{q^3} - a$, by writing $a$ using a basis for $\mathbb{F}_{q^3}$ over $\mathbb{F}_q$ and solving the resulting quadratic system using a Gröbner basis computation [11], or using the following much faster technique which exploits the fact that $B = 1$.

Since $B = 1$ is the only Bluher value for $k = 3$, writing $b = \frac{1}{q_0}(\gamma a - q_0) = u_0 a + v_0$ and $c = \frac{1}{q_0}(-q_0 a - \gamma q_0 + \gamma q_1 a) = u_1 a + v_1$ as in (5), the equation (8) becomes

$$(-a^q + u_0 a + v_0)^{q+1} - (-u_0 a^2 + (u_1 - v_0)a + v_1)^q = 0. \tag{10}$$

By setting $X_i = a^{q^i}$ we can rewrite this as $E = 0$ with

$$E := (-X_1 + u_0 X_0 + v_0)(-X_2 + u_0^q X_1 + v_0^q) - (-u_0^q X_1^2 + (u_1^q - v_0^q)X_1 + v_1^q)$$

where $E$ is considered as a polynomial in the $X_i$. Notice that the coefficient of $X_1^2$ vanishes, a consequence of $B = 1$. Raising (10) to the power $q$ while substituting $X_i^q$ with $X_{i+1}$ (and $X_3$ with $X_0$) yields the equation $\tilde{E} = 0$ with

$$\tilde{E} := (-X_2 + u_0^q X_1 + v_0^q)(-X_0 + u_0^{q^2} X_2 + v_0^{q^2}) - (-u_0^{q^2} X_2^2 + (u_1^{q^2} - v_0^{q^2})X_2 + v_1^{q^2}).$$

Since the factors $-X_2 + u_0^q X_1 + v_0^q$ coincide in $E$ and $\tilde{E}$ (essentially by construction) and $X_0$ occurs only in their cofactors, it is easy to see that $E + u_0\tilde{E} = \alpha_{12}X_1X_2 + \alpha_1 X_1 + \alpha_2 X_2 + \alpha$ with $\alpha_*$ being simple expressions in $u_0, u_1, v_0, v_1$. The special case $\alpha_{12} = 0$ can be handled easily so we assume that $\alpha_{12} \neq 0$ and rewrite the equation $\alpha_{12}X_1X_2 + \alpha_1 X_1 + \alpha_2 X_2 + \alpha = 0$ as

$$X_1 = -\frac{\alpha_2 X_2 + \alpha}{\alpha_{12}X_2 + \alpha_1}.$$

By raising this equation to the power $q$ (with the usual substitution of $X_i^q$ with $X_{i+1}$) one can express $X_2$ in terms of $X_0$, and after doing this again $X_0$ in terms of $X_1$. Substituting these three equations into each other gives an equation of the form $X_0 = \frac{\beta_2 X_0 + \beta}{\beta_{12}X_0 + \beta_1}$ or, equivalently,

$$\beta_{12}X_0^2 + (\beta_1 - \beta_2)X_0 - \beta = 0. \tag{11}$$

In the rare case that the l.h.s. of (11) is identically zero we abort the direct elimination method and apply the backup method; otherwise we check whether (11) has solutions in $\mathbb{F}_{2^{30}}$ and when it does so, we check whether they satisfy (10). It is not easy to classify which solutions of (11) lead to solutions of (10), although in practice almost all of them do so.

Experimentally we found that a randomly chosen $Q$ is eliminable by this method with probability very close to 1/2. A heuristic argument supporting this observation is that the derived equation (11) presumably behaves as a random degree 2 equation, which therefore has a solution in $\mathbb{F}_{2^{30}}$ with probability $\approx 1/2$.

Before explaining the backup method we first introduce three efficient techniques that we employed for solving related problems.

### 6.2 Finding a Degree 2 Elimination with Probability $q^{-2}$

A simple method for eliminating a degree 2 element $Q$ in $\mathbb{F}_{q^k}[X]$ consists of picking a random $a \in \mathbb{F}_{q^k}$, computing $b = u_0 a + v_0 = \frac{1}{q_0}(\gamma a - q_0)$ and $c = u_1 a + v_1 = \frac{1}{q_0}(-q_0 a - \gamma q_0 + \gamma q_1 a)$ as before, and then checking whether the triple $(a, b, c)$ leads by (7) to a Bluher value $B \in \mathcal{B}_k$ using Theorem 3, i.e., whether $P_k(\frac{1}{B}) = 0$ for the polynomials $P_i$ defined by (2). The success probability is expected to be about $q^{-3}$ per trial by Theorem 1.

A better success rate is achieved by guessing a root $r$ of the polynomial $X^{q+1} + aX^q + bX + c$ first (with $b$ and $c$ as above) and testing if the polynomial splits completely. Note that when $r$ is a root and we plug in $b$ and $c$, we have $r^{q+1} + ar^q + (u_0 a + v_0)r + (u_1 a + v_1) = 0$ from which we recover

$$a = -\frac{r^{q+1} + v_0 r + v_1}{r^q + u_0 r + u_1}\,.$$

From this we again compute $B = \frac{(b-a^q)^{q+1}}{(c-ab)^q} \in \mathbb{F}_{q^k}$ and check by Theorem 3 whether $B \in \mathcal{B}_k$. With this method the probability that the corresponding $B$ is a Bluher value has been increased (heuristically) to about $q^{-2}$ by [2, Thm. 5.6].

The recurrence (2) for computing $P_k(\frac{1}{B})$ as per Theorem 3 can be modified by

$$\begin{aligned} P_1^* &= 1,\ P_2^* = b - a^q, \\ P_i^* &= (b - a^q)^{q^{i-2}} P_{i-1}^* - (c - ab)^{q^{i-2}} P_{i-2}^*, \end{aligned}$$

where $P_k^*$ provides (after clearing denominators) a polynomial $P$ in $r$ having the same roots as $P_k(\frac{1}{B})$, and which may be used for the interpolation approach described next.

### 6.3 Using Interpolation to Find Roots of Featured Polynomials

When encountering the task of finding a root in $\mathbb{F}_{q^k}$ of a polynomial $P$ of high degree in one variable $X$ which can be written as a low degree polynomial in the variables $X, X^q, X^{q^2}, \ldots$, the method of picking random $r \in \mathbb{F}_{q^k}$ until $P(r) = 0$ can sometimes be sped up as follows.

Let $r_0, r_1, \ldots, r_\ell \in \mathbb{F}_{q^k}$ such that $r_1, \ldots, r_\ell$ are linearly independent over $\mathbb{F}_q$ (in particular $\ell \leq k$) and let $\mathcal{R} := \{r_0 + \sum_{i=1}^{\ell} c_i r_i \mid c_1, \ldots, c_\ell \in \mathbb{F}_q\}$. Then there is a low degree polynomial $\tilde{P}$ satisfying $\tilde{P}(c_1, \ldots, c_\ell) = P(r_0 + \sum_{i=1}^{\ell} c_i r_i)$; let $D = \deg(\tilde{P})$ and assume $D < q$. The polynomial $\tilde{P}$ can be computed by evaluating $P$ at $\binom{D+\ell}{\ell}$ elements of $\mathcal{R}$ and interpolating, after which the $q^\ell$ values $P(r)$ for $r \in \mathcal{R}$ can be obtained by evaluating $\tilde{P}$. Notice that still $q^\ell$ operations are needed to cover the set $\mathcal{R}$ but if evaluating $P$ is sufficiently more expensive than evaluating $\tilde{P}$, this method is faster. The optimal value of $\ell$ depends on $q$, $D$ and the evaluation costs for the two polynomials.

### 6.4 Using GCD Computations

In certain situations we have to find an element in $\mathbb{F}_{q^k}$ which is a root of two polynomials $P_1$ and $P_2$. In this case one can speed up the interpolation approach as follows.

Let $\tilde{P}_1$ and $\tilde{P}_2$ be the low degree polynomials corresponding to $P_1$ and $P_2$ and let their degrees be $D_1$ and $D_2$, respectively. If a tuple $(c_1, \ldots, c_\ell)$ leads to a root of $P_1$ and $P_2$ then $c_1$ is a root of $\tilde{P}_i(c_1, \ldots, c_\ell)$, $i = 1, 2$, considered as a polynomial in the variable $c_1$. Therefore $c_1$ is also a root of the greatest common divisor of these two univariate polynomials. If $q^k$ is much bigger than $D_1 D_2$, the degree of the GCD is usually not bigger than 1 so that one trades $q$ evaluations of $\tilde{P}_1$ (and/or $\tilde{P}_2$) for one GCD computation.

### 6.5 Backup Method

When the direct method fails, which occurs with probability $\approx 1/2$, we use the following backup approach, based on the idea from [11]; for the remainder of this section we assume $\mathbb{F}_{q^k} = \mathbb{F}_{2^{30}}$. Instead of using $q = 2^{10}$, $k = 3$ we use $q = 2^6$, $k = 5$, which by Theorem 1 means there are $q^2 + 1 = 4097$ Bluher values, but now the r.h.s. of the relevant equation will have higher degree and thus a smaller chance that the cofactor is 1-smooth.

Let $y = x^{1024}$ and $\overline{x} = x^{16}$, so that $y = \overline{x}^{64}$. As $x^{1025} + x + \gamma = 0$ we have $x = \frac{\gamma}{y+1}$ and $\overline{x} = \left(\frac{\gamma}{y+1}\right)^{16}$. We therefore have

$$\begin{aligned}\overline{x}^{65} + a\overline{x}^{64} + b\overline{x} + c &= y\left(\tfrac{\gamma}{y+1}\right)^{16} + ay + b\left(\tfrac{\gamma}{y+1}\right)^{16} + c \\ &= \tfrac{1}{(y+1)^{16}}\left(ay^{17} + cy^{16} + (\gamma^{16} + a)y + (b\gamma^{16} + c)\right). \end{aligned} \tag{12}$$

Now let $\widetilde{Q}(X) := (X+1)^2 Q\left(\frac{\gamma}{X+1}\right)$ so that $Q(x) = \frac{\widetilde{Q}(y)}{(y+1)^2}$, and consider the lattice

$$L_{\widetilde{Q}} := \left\{(w_0, w_1) \in \mathbb{F}_{q^5}[X]^2 \mid w_0 + \tfrac{(X+1)^{16}}{\gamma^{16}} w_1 \equiv 0 \pmod{\widetilde{Q}}\right\}.$$

In general, $L_{\widetilde{Q}}$ has a basis of the form $(X + u_0, u_1), (v_0, X + v_1)$ with $u_i, v_i \in \mathbb{F}_{q^5}$. Thus for $a \in \mathbb{F}_{q^5}$ we have $(X + u_0 + av_0, aX + u_1 + av_1) \in L_{\widetilde{Q}}$. Substituting this element into $w_0, w_1$ and evaluating the resulting expression at $y$ leads to the r.h.s. of (12) being

$$\tfrac{1}{(y+1)^{16}}\left(ay^{17} + (u_1 + av_1)y^{16} + (\gamma^{16} + a)y + \gamma^{16}(u_0 + av_0) + u_1 + av_1\right)$$

and thus $b = av_0 + u_0$ and $c = av_1 + u_1$. The l.h.s. of (12) transforms into a Bluher polynomial provided that $(a^{64} + b)^{65} = B(ab + c)^{64}$ for some $B \in \mathcal{B}_5$. This results in the equation

$$(a^{64} + v_0 a + u_0)^{65} + B(v_0 a^2 + (u_0 + v_1)a + u_1)^{64} = 0\,. \tag{13}$$

As before, $\mathbb{F}_{q^5}$-roots of (13) can be computed, if they exist, via a GCD computation or using a basis for $\mathbb{F}_{q^5}$ over $\mathbb{F}_q$ and solving the resulting quadratic system using a Gröbner basis approach. Instead of solving (13) directly, we used the probability $q^{-2}$ method and interpolation with $\ell = 3$ so that we expect to find about $q = 2^6$ completely splitting l.h.s. of (12) per interpolation.

For each of these we check whether the r.h.s. of (12) is 2-smooth as follows. Denote by $R$ the polynomial corresponding to its numerator and let $m$ be a linear fractional transformation over $\mathbb{F}_{2^{60}}$ mapping the two roots of $\widetilde{Q}$ to 0 and $\infty$. Then the polynomial $R$ becomes (up to a scalar) $X^{16} - \alpha X$ for some $\alpha \in \mathbb{F}_{2^{60}}^\times$, and the transformation $\overline{m}$ maps $R$ to $X^{16} - \overline{\alpha}X$, where $\bar{\cdot}$ denotes the $\mathbb{F}_{2^{60}}/\mathbb{F}_{2^{30}}$ Galois conjugate, i.e., powering by $2^{30}$. If and only if $\alpha \in \mathbb{F}_{2^{60}}$ is a fifteenth power, does the r.h.s. of (12) split completely over $\mathbb{F}_{2^{60}}$ and is thus 2-smooth over $\mathbb{F}_{2^{30}}$. As $\widetilde{Q}$ is irreducible, the transformation $\overline{m}^{-1}m$ exchanges 0 and $\infty$ and is thus of the form $X \mapsto \frac{\beta}{X}$ for some $\beta \in \mathbb{F}_{2^{60}}^\times$, while mapping $X^{16} - \overline{\alpha}X$ (up to a scalar) to $X^{16} - \alpha X$ and hence $\alpha\overline{\alpha} = \beta^{15}$, which implies that $\alpha = \alpha^{2+2^{30}}\beta^{-15}$ is already a third power. Therefore, we expect the r.h.s. of (12) to be 2-smooth with probability $\frac{1}{5}$, which is much higher than for a random polynomial of this degree, thanks to it being transformable to a Bluher polynomial.

Furthermore, whenever the r.h.s. is 2-smooth it must factorise into a product of five linear polynomials and five degree two polynomials. In particular, assume that the polynomial $R$ on the r.h.s. of (12) splits completely over $\mathbb{F}_{2^{60}}$. Similarly as above, denote by $m$ a linear fractional transformation mapping $R$ to $X^{16} - X$ and the two roots of $\widetilde{Q}$ to 0 and $\infty$. Then the fifteen other roots of $R$ are $r_i = m(\zeta_{15}^i)$, $i = 0, \ldots, 14$, with $\zeta_{15} \in \mu_{15}$ a fixed primitive fifteenth root of unity. Notice that $\overline{\zeta_{15}} = \zeta_{15}^4$. In order to find the roots $r_i$ contained in the subfield $\mathbb{F}_{2^{30}}$ one has to solve $m(\zeta_{15}^i) = \overline{m(\zeta_{15}^i)} = \overline{m}(\zeta_{15}^{4i})$ or equivalently $\overline{m}^{-1}m(\zeta_{15}^i) = \zeta_{15}^{4i}$; let $n = \overline{m}^{-1}m$. Since $n$ exchanges 0 and $\infty$, and because $n\overline{n} = I_2$ holds, $n$ is actually a transformation of the form $X \mapsto \frac{b}{X}$ for some $b \in \mathbb{F}_{2^{30}}^\times$. Furthermore $n$ maps $X^{16} - X$ to itself which implies $n(\mu_{15}) \subset \mu_{15}$,

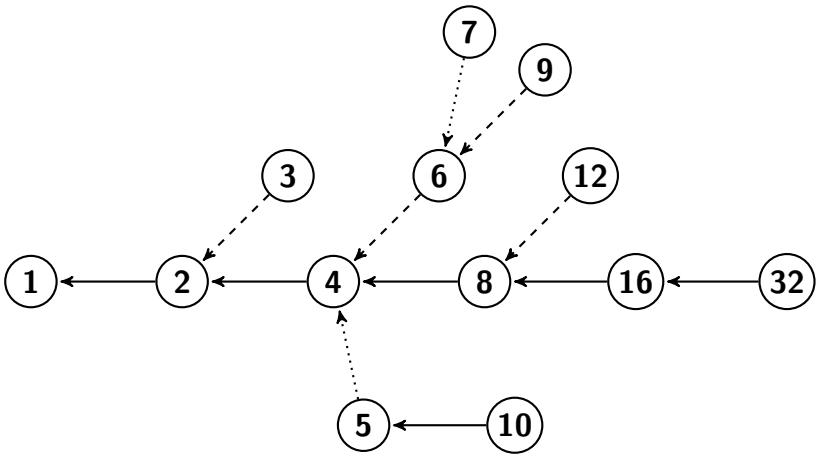



**Fig. 1.** Overview of the small-degree, 'non-classical' descent methods. The encircled numbers represent the degrees of elements. Solid arrows indicate two-to-one descents (including the backup strategy for $2 \to 1$), dashed arrows depict three-to-two descents, and dotted arrows represent specific odd-degree methods.

hence $b \in \mu_{15} \cap \mathbb{F}_{2^{30}} = \mu_3$. Therefore the equation $n(\zeta_{15}^i) = \zeta_{15}^{4i}$ from above becomes $\zeta_{15}^{5i} = b$ which has exactly five solutions. Thus the cofactor of $\widetilde{Q}$ splits into five polynomials of degree one and five of degree two. We remark that the same reasoning can be applied to determine the possible splitting pattern for a general Bluher polynomial in positive characteristic.

For each set of five degree two polynomials, we attempt to eliminate each of them using the direct method, which succeeds with probability $1/2^5$, assuming these eliminations are independent. One thus expects to eliminate such degree two elements after trying 160 Bluher values, a figure which was borne out by our experiments. One also expects the approach to fail for a given irreducible of degree two with probability $(1-\frac{1}{160})^{4097} \approx 7 \cdot 10^{-12}$. If this happened, we simply restarted the elimination at an ancestor with a different seed for randomness. In terms of timings per degree two elimination, on average these ranged between 1.1ms on some machines and just less than 1ms on others.

## 7 Small Even Degree Elimination Techniques

An overview of the various descent methods that we applied for the smaller degrees is depicted in Figure 1. In this short section we explain those methods for even degrees that are based on degree 2 elimination, and in the next section we detail the methods for odd degrees.

### 7.1 Degree 4 Elimination

By Proposition 1 this case is reduced to the elimination of a degree 2 polynomial over $\mathbb{F}_{q^6} = \mathbb{F}_{2^{60}}$. We use the probability $q^{-2}$ method and interpolation with $\ell = 1$ to obtain a polynomial $\tilde{P}$ of degree 15 over $\mathbb{F}_{q^6}$ and search for its zeroes in $\mathbb{F}_q$ as follows. By choosing an $\mathbb{F}_q$ basis of $\mathbb{F}_{q^6}$ the polynomial $\tilde{P}$ can be expressed as a linear combination of six polynomials $\tilde{P}_1, \ldots, \tilde{P}_6 \in \mathbb{F}_q[X]$ so that an $\mathbb{F}_q$-zero of $\tilde{P}$ is also a zero of $\mathrm{GCD}(\tilde{P}_1, \ldots, \tilde{P}_6)$. In the rare case that the degree of this GCD is bigger than 1 we evaluate $\tilde{P}$ at all elements of $\mathbb{F}_q$. Usually it is sufficient to compute the GCD of at most three of the $\tilde{P}_i$.

### 7.2 Degree $2d$ to Degree $d$

For $2d \in \{8, 10, 16, 32\}$ we rewrite an irreducible in $\mathbb{F}_{q^3}[X]$ of degree $2d$ into factors of degree (at most) $d$, using the algorithm outlined in the proof sketch of Proposition 1. This method is based on degree 2 elimination over the fields $\mathbb{F}_{q^{3d}}$, which we perform using the technique already described in §6.2, since the subsequent costs dominate the elimination costs by a large factor.

## 8 Small Odd Degree Elimination Techniques

In this section we explain the methods employed for eliminating irreducible polynomials in $\mathbb{F}_{q^k}[X]$ of small odd degree $d$ into polynomials of degrees $d-1, \ldots, 1$. In the same way that the degree 2 elimination extends to any even degree polynomial these techniques allow to eliminate irreducible polynomials of degree $nd$ into polynomials of degrees $n(d-1), \ldots, n$.

### 8.1 Degree 3 Elimination

We show in this section how the degree 2 elimination technique can be extended to eliminate degree 3 polynomials. Instead of eliminating degree 2 polynomials into linear polynomials, we eliminate degree 3 polynomials into degree 2 polynomials: it is a 3–to–2 elimination.

As in §3 we use the field representation $\mathbb{F}_{q^{kn}} = \mathbb{F}_{q^k}[X]/(I)$, where $I$ divides $h_1 X^q - h_0$. We may assume that $h_0$ and $h_1$ have degree at most 1, since in our setup $h_0 = X + \gamma$ and $h_1 = X$. The 3–to–2 elimination takes as input a cubic polynomial $Q$ in $\mathbb{F}_{q^k}[X]$ and rewrites it into a product of linear and quadratic polynomials $Q_i \in \mathbb{F}_{q^k}[X]$ such that $Q \equiv \prod_i Q_i \bmod I$.

In our setup, the base case consists in eliminating cubic polynomials over $\mathbb{F}_{2^{30}}$ into quadratic and linear polynomials over $\mathbb{F}_{2^{30}}$, so $q = 2^{10}$ and $k = 3$. As mentioned, this technique allows to eliminate irreducible polynomials of degree $3d$ over $\mathbb{F}_{2^{30}}$ into polynomials of degree $2d$ and $d$ over $\mathbb{F}_{2^{30}}$ in the same way that the degree 2 elimination extends to any even degree polynomials. Concretely, let $Q \in \mathbb{F}_{2^{30}}[X]$ be an irreducible polynomial of degree $3d$. Then $Q$ splits in $\mathbb{F}_{2^{30d}}[X]$ into $d$ Galois-conjugate irreducible factors, each of degree 3. Let $\hat{Q}$ be any of these factors. Let $N\colon \mathbb{F}_{2^{30d}}[X] \to \mathbb{F}_{2^{30}}[X]$ be the norm map. Then $Q = N(\hat{Q})$. Applying the degree 3 elimination to $\hat{Q}$ yields a product $\prod_i \hat{Q}_i$ such that each $\hat{Q}_i$ is linear or quadratic in $\mathbb{F}_{2^{30d}}[X]$, and $\hat{Q} \equiv \prod_i \hat{Q}_i \bmod I$. Thus each $Q_i = N(\hat{Q}_i)$ is of degree $d$ or $2d$ in $\mathbb{F}_{2^{30}}[X]$, and $Q \equiv \prod_i Q_i \bmod I$.

This method is used in our computation to eliminate polynomials of degree 6, 9 and 12.

**Degree 3 to Degree 2 Elimination** Let $k \geq 3$. We now show how to eliminate a cubic polynomial $Q$ over $\mathbb{F}_{q^k}$ into a product of linear and quadratic polynomials over $\mathbb{F}_{q^k}$. As previously, let $\mathcal{B}_k$ be the set of all values $B \in \mathbb{F}_{q^k}$ such that $X^{q+1} - BX + B$ splits completely in $\mathbb{F}_{q^k}[X]$. Any polynomial of the form $X^{q+1} + aX^q + bX + c$ with $c \neq ab$ and $b \neq a^q$ splits completely in $\mathbb{F}_{q^k}[X]$ whenever $B = \frac{(b-a^q)^{q+1}}{(c-ab)^q}$ is in $\mathcal{B}_k$. Consider the polynomial

$$H_0(X, Y) = XY + aY + bX + c \in \mathbb{F}_{q^k}[X, Y]\,,$$

and define the transformation

$$T_\delta\colon X \longmapsto \frac{X^2 + X + \delta}{X + 1}\,.$$

Let $H_\delta(X, Y) = H_0(T_\delta(X), T_{\delta^q}(Y))$. Whenever $B = \frac{(b-a^q)^{q+1}}{(c-ab)^q} \in \mathcal{B}_k$, the numerator of $H_\delta(X, X^q)$ splits into linear and quadratic polynomials in $R_1 = \mathbb{F}_{q^k}[X]$. The denominator of $H_\delta$ equals $(X+1)(Y+1)$, and the polynomial $(X+1)(Y+1)H_\delta$ is of degree 2 in $Y$. Therefore the image of $(X+1)(Y+1)H_\delta$ in $R_2 = \mathbb{F}_{q^k}[X][1/h_1]$ has denominator $h_1^2$, so to get rid of all the denominators, we will work with the polynomial

$$G_\delta = h_1^2(X + 1)(Y + 1)H_\delta\,.$$

On one hand, we want $G_\delta$ to split into linear and quadratic polynomials in $R_1$, i.e., $B$ should be an element of $\mathcal{B}_k$. On the other hand, we want $Q$ to divide the image of $G_\delta$ in $R_2$. The latter is of degree at most $2 + 2\max(\deg h_0, \deg h_1)$. When $h_0$ and $h_1$ are of degree 1, and $Q$ divides the image of $G_\delta$ in $R_2$, the cofactor is of degree at most 1.

We shall now describe how to find suitable values of $\delta, a, b$ and $c$ in $\mathbb{F}_{q^k}$. First, observe that $G_\delta$ is a linear combination of the polynomials

$$\begin{aligned} I_0 &= h_1^2(X+1)(Y+1)T_\delta(X)T_{\delta^q}(Y), \\ J_0 &= h_1^2(X+1)(Y+1)T_{\delta^q}(Y), \\ K_0 &= h_1^2(X+1)(Y+1)T_\delta(X), \\ L_0 &= h_1^2(X+1)(Y+1), \end{aligned}$$

as $G_\delta = I_0 + aJ_0 + bK_0 + cL_0$. Let $I, J, K$ and $L$ be the reductions modulo $Q$ of the images in $R_2$ of $I_0, J_0, K_0$ and $L_0$ respectively; in particular, these are polynomials in $R_2$ of degrees at most 2. Then, $Q$ divides $G_\delta$ in $R_2$ if and only if $I + aJ + bK + cL = 0$. The latter is a linear system for the three variables $a, b$ and $c$, in the 3-dimensional vector space of polynomials of degree 2. Solving this linear system with $\delta$ as a symbolic variable yields a solution $a, b, c \in \mathbb{F}_{q^k}[\delta]$. Choosing random values for $\delta$, the resulting $B$ is expected to fall in $\mathcal{B}_k$ with an expected probability of $q^{-3}$. For $k = 3$, there is only one element in $\mathcal{B}_k$, so it might happen that no good value of $\delta$ exists. In that case, we start again with the transformation

$$T_{\delta,\alpha} \colon X \longmapsto \frac{X^2 + X + \delta}{X + \alpha},$$

in place of $T_\delta$, for random values $\alpha \in \mathbb{F}_{q^k}$.

**Optimising the Elimination** Suppose we have computed $a, b, c \in \mathbb{F}_{q^k}[\delta]$, and are looking for values of $\delta$ that give rise to a Bluher value. Instead of trying random values for $\delta$, one can directly find these which yield a $B$ in $\mathcal{B}_k$ via the characteristic polynomial of inverse Bluher values as per Theorem 3. By using the recurrence in §6.2 and interpolation (cf. §6.3, in our practical setting we use $\ell = 1$) the random guessing approach could be sped up considerably.

### 8.2 Eliminating Polynomials of Degree 5 and Degree 7

Let $Q$ be a polynomial of degree $d$. Consider two polynomials $F = \sum_{i=0}^{d_F} f_i X^i$ and $G = \sum_{i=0}^{d_G} g_i X^i$ in $\mathbb{F}_{q^k}[X]$ with $d_F, d_G < d$. We have

$$F^q G - F G^q \equiv \frac{1}{h_1^{d-1}} \Big( F^{(q)}\big(\tfrac{h_0}{h_1}\big) h_1^{d-1} G - F G^{(q)}\big(\tfrac{h_0}{h_1}\big) h_1^{d-1} \Big) \pmod{I},$$

with the numerator of the r.h.s. of degree at most $2d-2$. The coefficients of this numerator are simple polynomials in the $f_i$, $f_i^q$, $g_i$ and $g_i^q$, in particular, they are linear in each of these variables. If the polynomial $G$ is fixed, it is easy to find all polynomials $F$ such that the r.h.s. is zero modulo $Q$. Indeed, for an $\mathbb{F}_q$-basis $(\alpha_i)$ of $\mathbb{F}_{q^k}$ one can write $f_i = \sum_{j=1}^{k} f_{ij}\alpha_j$ with $f_{ij} \in \mathbb{F}_q$ (implying $f_i^q = \sum_{j=1}^{k} f_{ij}\alpha_j^q$) and reducing the r.h.s. modulo $Q$ to obtain a linear system of $dk$ equations in the $dk$ unknowns $f_{ij}$. Since $F = G$ is always a solution, the corresponding matrix has rank at most $dk-1$; if its rank is smaller, a non-trivial pair $(F, G)$ exists which leads to an elimination of $Q$ into polynomials of degrees smaller than $d$.

Heuristic arguments suggest that the probability of finding a non-trivial pair $(F, G)$ for a randomly chosen $G$ is about $q^{-2}$. Namely, discarding the $1 + (q+1)(q^{dk}-1)$ trivial solutions $\{(F, G) \mid c_F F = c_G G \text{ for some } c_F, c_G \in \mathbb{F}_q\}$, one expects that about a $q^{-dk}$ part of the roughly $q^{2dk}$ remaining pairs $(F, G)$ give rise to a r.h.s. divisible by $Q$. Since one non-trivial solution $(F, G)$ entails $q^2-q$ non-trivial solutions $(aF + bG, G)$, with $a, b \in \mathbb{F}_q$, $a \neq 0$, the probability of finding a non-trivial solution for a fixed $G$ is heuristically about $q^{-2}$.

The cost of the algorithm as presented is $O(q^2(dk)^3)$ (with a lower exponent if fast matrix multiplication is used; since the subsequent costs of the elimination dominate, we did not do

this). If $q$ is big compared to $dk$, the following variants might be advantageous. If the rank of the matrix corresponding to the linear system obtained from a randomly chosen $G$ has rank at most $dk-2$ then all its minors vanish. Thus, instead of directly computing the rank of the matrix, one can check whether a few minors vanish and, if so, compute the rank. Since the minors are polynomials of degree $dk-1$ in the $g_{ij} \in \mathbb{F}_q$ defined by $g_i = \sum_{j=1}^{k} g_{ij}\alpha_j$, the interpolation approach as well as the GCD approach may be used. For an $\ell = 1$ interpolation the cost is $O(q(dk)^4)$ and for $\ell = 2$ it is $O((dk)^5 + q(dk)^2)$.

In our computation we used this method for degrees 5 and 7 for which the $\ell = 2$ interpolation was optimal ($q = 1024$ and $dk = 15, 21$). We remark that the subsequent costs for this method are much bigger than for the Gröbner basis method, but this was not a significant concern as the resulting costs for these degrees were relatively inexpensive.

## 9 Computing Roots of Bluher Polynomials and Their Transformations

In order to find all roots of a Bluher polynomial or a transformation thereof, which is known to split completely, we use two different methods.

If a transformation is known which maps the polynomial to an explicit polynomial with known roots, this transformation is used to obtain the roots. This applies to the following cases:

- the direct method in degree 2 and degree 3 elimination, where we transform to the only Bluher polynomial $X^{q+1} - X + 1$ and use its pre-computed roots;
- the backup method in degree 2 elimination for which the r.h.s. polynomial can be transformed to $X^{16} - \beta^{15}X$;
- and the degree $nd$ to $n(d-1), \ldots, n$ elimination where the splitting of the polynomial is obvious.

In the other cases of degree $2^t$ elimination, one root of the polynomial is known so that by sending this root to $\infty$ we have to find the roots of a polynomial $\tilde{P} = X^q + a_1X + a_0$. The case $a_1 = 0$ is trivial so we assume that $\tilde{P}$ has no multiple roots. Let $r_0, r_1 \in \mathbb{F}_{q^k}$ be random elements and compute $s_1X + s_0 \equiv \sum_{i=0}^{k-1}(r_1X + r_0)^{q^i} \pmod{\tilde{P}}$. By construction $(s_1X + s_0)^q \equiv s_1X + s_0 \pmod{\tilde{P}}$ holds so that $\tilde{P}$ divides $(s_1X + s_0)^q - (s_1X + s_0)$. If $s_1$ does not vanish, the roots of $\tilde{P}$ are $\frac{\beta - s_0}{s_1}$, $\beta \in \mathbb{F}_q$; otherwise one tries other random pairs $(r_0, r_1)$.

Finally, for the degree 6, 9, 10 and 12 eliminations, the polynomials are also a transformation of a Bluher polynomial. Three roots of the Bluher polynomial are found with the Cantor–Zassenhaus algorithm, from which all the roots of the original polynomial are recovered.

## 10 Classical Descent

As stated in the introduction, we set ourselves a discrete logarithm challenge using the digit expansion (to the base $2^{30}$) of the mathematical constant $\pi$, which is viewed, as is common, as a source of pseudorandomness. Specifically, the target element $h_\pi \in \mathbb{F}_{2^{30750}}$ is represented by a polynomial over $\mathbb{F}_{2^{30}}$ of degree 1024.

In the individual logarithm phase, or the 'descent' phase, of the index calculus method, we seek to rewrite this polynomial, modulo $X^{1025} + X + \gamma$, step-by-step as a product of polynomials having smaller degrees, until we have expressed the target element as a product of factor base elements.

### 10.1 Descent Analysis

Performing a descent starting from degree 1024 constitutes a substantial challenge. In order to optimise the overall computational cost we employ a bottom-up approach in our analysis. This means that for every degree $d = 2, 3, 4, \ldots$ we determine the expected cost of computing a

logarithm of an element represented by a polynomial of degree $d$. This cost is the sum of finding a (good) representation as a product of lower degree polynomials (the 'direct cost'), plus the expected total cost of the factors appearing in the product, based on the earlier cost estimations for degrees $< d$ (the 'subsequent cost').

When computing the expected direct and subsequent costs, one needs statistical information about the degree pattern distribution and the resulting cost distribution for a random polynomial over $\mathbb{F}_{2^{30}}$ of a given degree. These can be obtained using formulas for the number of irreducible polynomials and an approach involving formal power series. As the analysis becomes more difficult when the degrees increase, we round the input costs appropriately and sometimes replace the field $\mathbb{F}_{2^{30}}$ by the field $\mathbb{F}_2$ (without much loss of accuracy). Assisted by Magma for the power series computations, we list in Table 2 the expected costs for each of the degrees up to 40.[8]

**Table 2.** Rounded cost estimations used in the descent phase. The cost unit is normalised to the equivalent of one degree 16 elimination.

| deg | cost | method and degrees |
|---|---|---|
| 2 | 0 | two-to-one: $\rightsquigarrow$ 1 |
| 3 | 0 | three-to-two: $\rightsquigarrow$ 2, 1 |
| 4 | 0 | two-to-one: $\rightsquigarrow$ 2 |
| 5 | 0 | five-to-four: $\rightsquigarrow$ 4, 3, 2, 1 |
| 6 | 0 | three-to-two $\rightsquigarrow$ 4, 2 |
| 7 | 0 | seven-to-six: $\rightsquigarrow$ 6, 5, 4, 3, 2, 1 |
| 8 | 0 | two-to-one: $\rightsquigarrow$ 4 |
| 9 | 0.3 | three-to-two: $\rightsquigarrow$ 6, 3 |
| 10 | 0.3 | two-to-one: $\rightsquigarrow$ 5 |
| 11 | 10 | classical: $\rightsquigarrow$ 86, 70 |
| 12 | 0.5 | three-to-two: $\rightsquigarrow$ 8, 4 |
| 13 | 44 | classical: $\rightsquigarrow$ 100, 71 |
| 14 | 120 | classical: $\rightsquigarrow$ 99, 71, or 115, 72 |
| 15 | 160 | classical: $\rightsquigarrow$ 114, 72 |
| 16 | 1 | two-to-one: $\rightsquigarrow$ 8 |
| 17, 18 | 250 | classical: $\rightsquigarrow$ 73, 120 |
| 19, 20 | 360 | classical: $\rightsquigarrow$ 81, 119 |
| 21, 22 | 480 | classical: $\rightsquigarrow$ 89, 118 |
| 23, 24 | 630 | classical: $\rightsquigarrow$ 97, 117 |
| 25, 26 | 790 | classical: $\rightsquigarrow$ 105, 116 |
| 27, 28 | 960 | classical: $\rightsquigarrow$ 113, 115 |
| 29, 30 | 1150 | classical: $\rightsquigarrow$ 121, 114 |
| 31 | 1350 | classical: $\rightsquigarrow$ 129, 113 |
| 32 | 1000 | two-to-one: $\rightsquigarrow$ 16 |
| 33, 34 | 1580 | classical: $\rightsquigarrow$ 137, 112 |
| 35, 36 | 1830 | classical: $\rightsquigarrow$ 145, 111 |
| 37, 38 | 2000 | classical: $\rightsquigarrow$ 116, 147 |
| 39, 40 | 2300 | classical: $\rightsquigarrow$ 122, 148 |

### 10.2 Initial Split and Classical Descent

Having finished the bottom-up analysis, starting from lowest degrees, the actual descent computation was performed top-down, starting with the target element of degree 1024.

In the first phase, referred to as 'initial split', the target element $\beta$ is rewritten as a fraction

$$g^i h_\pi = r(x)/s(x),$$

[8] In common descent algorithms, a smaller degree usually means a lower cost, which however is not generally true in our case. So in fact, our bottom-up analysis starts with the degrees for which the non-classical descent methods apply, cf. Fig. 1.

where $\deg r + \deg s = 1024$ and the integer $i$ should be chosen so that the polynomials $r, s \in \mathbb{F}_{2^{30}}[X]$ have a favourable factorisation pattern. The descent analysis suggested that a split with unbalanced degrees is preferable, so we searched for splittings with $\deg r = 205$ and with $\deg r = 256$. The most promising initial split we found was using $\deg r = 205$, $\deg s = 819$ and where $i = 47\,611\,005\,802$, resulting in a factorisation with largest irreducible factor being of degree 672.

Next the 'classical descent' was performed, which is based on the following rewriting method. Choose some $a \in \{0, 1, \ldots, 10\}$ and define $\overline{x} := x^{2^{10-a}}$ and $y := \overline{x}^{2^a}$, so that $\overline{x} = (\gamma/(y{+}1))^{2^{10-a}}$. Then for polynomials $u, v \in \mathbb{F}_{2^{30}}[X]$ we have

$$u(\overline{x}^{2^a})\overline{x} + v(\overline{x}^{2^a}) = u(y)\big(\tfrac{\gamma}{y+1}\big)^{2^{10-a}} + v(y).$$

If a target field element represented by a polynomial $Q \in \mathbb{F}_{2^{30}}[X]$ of degree $d$ is to be eliminated, we choose the polynomials $u, v$ such that $Q$ divides one of the sides. Now balancing the degrees such that $\deg u + \deg v = d + e$ (for $e \in \{0, 1\}$) gives $2^{30(e+1)}$ trials for obtaining a smooth cofactor and a smooth other side.

We refer to Table 3 for computational details of the initial split and the classical descent. This phase resulted in a number of polynomials to be eliminated by the non-classical descent methods, the timings of which are summarised in Table 4.

**Table 3.** Initial split and classical descent: schedule and running times.

| is | when | degree | #polys | iterations | core hours |
|---|---|---|---|---|---|
| | 09 Sep - 20 Sep 2016 | 1024 | 1 | 60 G | 248 140 |

| cd | when | degree | #polys | iterations | core hours |
|---|---|---|---|---|---|
| 13 | Oct - 30 Oct 2016 | 672 | 1 | 80 G | 220 406 |
| 31 | Oct - 02 Dec | 466 | 1 | 80 G | 395 216 |
| 08 | Dec - 03 Jan | 130 | 1 | 80 G | 121 988 |
| 17 | Dec - 26 Dec | 106 | 1 | 40 G | 120 821 |
| 14 | Jan - 22 Jan 2017 | 64 | 1 | 40 G | 85 721 |
| 22 | Jan - 28 Jan | 45 | 1 | 26 G | 52 127 |
| 28 | Jan - 01 Feb | 40 | 1 | 32 G | 56 721 |
| 01 | Feb - 02 Feb | 38 | 1 | 16 G | 26 620 |
| 03 | Feb - 04 Feb | 35 | 1 | 16 G | 23 298 |
| 04 | Feb - 07 Feb | 34 | 3 | 48 G | 69 144 |
| 08 | Feb - 11 Feb | 31 | 3 | 48 G | 70 381 |
| 11 | Feb - 14 Feb | 30 | 3 | 96 G | 143 208 |
| 18 | Feb - 21 Feb | 29 | 2 | 32 G | 47 987 |
| 22 | Feb - 04 Mar | 28 | 4 | 64 G | 96 933 |
| 05 | Mar - 08 Mar | 27 | 3 | 48 G | 72 982 |
| 10 | Mar - 13 Mar | 26 | 2 | 32 G | 38 565 |
| 14 | Mar - 18 Mar | 25 | 4 | 40 G | 43 820 |
| 18 | Mar - 26 Mar | 24 | 8 | 100 G | 110 932 |
| 29 | Mar - 01 Apr | 23 | 5 | 60 G | 59 165 |
| 01 | Apr - 09 Apr | 22 | 9 | 108 G | 108 868 |
| 11 | Apr - 24 Apr | 21 | 12 | 173 G | 153 994 |
| 24 | Apr - 08 May | 20 | 12 | 216 G | 192 121 |
| 12 | May - 30 May | 19 | 19 | 486 G | 347 169 |
| 31 | May - 09 Jul | 18 | 43 | 1143 G | 788 765 |
| 19 | Jul - 02 Aug | 17 | 39 | 879 G | 588 148 |
| 05 | Aug - 11 Oct | 15 | 116 | 2761 G | 1 780 136 |
| 14 | Oct - 19 May 2018 | 14 | 209 | 6558 G | 4 049 638 |
| 20 | May - 10 Sep | 13 | 592 | 6961 G | 4 361 593 |
| 10 | Sep - 23 Jan 2019 | 11 | 1687 | 4647 G | 2 851 369 |
| | *total running time* | | | | 17 077 836 |

**Table 4.** Timings of the non-classical descent methods.

| degree | #polys | core hours |
|---|---|---|
| 32 | 8 | 2 800 053 |
| 16 | 8622 | 2 944 153 |
| 12 | 5644 | 976 533 |
| 10 | 4694 | 790 316 |
| 9 | 4184 | 372 762 |
| 8 | 3998 | 1388 |
| 7 | 3671 | 236 195 |
| 6 | 3467 | 615 |
| 5 | 3384 | 719 |
| 3 | 3744 | 8 |
| 0,1,2,4 | 22 973 | 2 |
| *total* | | 8 122 744 |

## 11 Reflections on the Security of the Proposal of Canteaut *et al.*

One question which has a direct bearing on the security of the secure compressed encryption scheme of Canteaut *et al.* [6], is what is the complexity of solving discrete logarithms in the fields $\mathbb{F}_{2^n}$ with $n$ prime and $\approx 16\,000$?

In fact $n \approx 10\,322 > 2^{80/6}$ would suffice for their security condition, which is that the first stage of index calculus – which has complexity $O((2^{\log_2 n})^6)$ when using the approach of Joux-Pierrot – should have 80-bit security. Firstly, one can reduce this slightly by using a degree $n$ irreducible factor of $h_1(X^q)X - h_0(X^q)$ with $\deg(h_1) = 2$ and $\deg(h_0) = 1$, allowing $q = 2^{13}$ to be used rather than the stipulated $2^{14}$. Secondly, ignoring the descent is an overly conservative approach as one is then effectively assuming that logarithms can be computed in polynomial time, which partly explains why the performance of the scheme is so slow. Thirdly, one can instead use $q = 2^{12}$ and $h_1 := X^3 + X^2 + X + 1$, $h_0 := X^5 + X^4 + X^2$ as $h_1(X^q)X - h_0(X^q)$ has an irreducible factor of degree 10 333, which is the next prime after 10 322. For such $h_1, h_0$, the analysis of Joux-Pierrot does not apply in its entirety, but the alternative used by Kleinjung for the computation of discrete logarithms in $\mathbb{F}_{2^{1279}}$ [25] gives an $O(q^7)$ algorithm for computing the logarithms of all irreducible elements of $\mathbb{F}_{q^{10333}}$ of degree $\leq 4$.

Given that a target discrete logarithm is in $\mathbb{F}_{2^{10333}}$, any irreducible elements of degree $d$ obtained during the descent can be split into some degree $d/\mathrm{GCD}(d, 12)$ irreducibles over $\mathbb{F}_q$ (or a subfield). Hence for the initial split one can expect to obtain only elements of degree, say $< 200$, in a reasonable time. Furthermore, since the logarithms of elements of degree up to 4 are known, $d/\mathrm{GCD}(d, 12)$ need only be a small power of 2 times 1, 2, 3 or 4 in order to apply the GKZ step, so that, e.g., $d = 288$, 384, 576 or 768 are also favourable degrees to search for during the initial split. Note that since the degrees of $h_i$ are larger than in the present case, finding cofactors that are 1-smooth will be more costly for each elimination. However, whether an individual logarithm can be computed in this field more efficiently than the factor base logarithms would require a detailed model and analysis of both parts, and we leave this as an open, but not particularly pressing question.

More generally, finding the optimal field representation, factor base logarithm method, and descent strategy for any given extension degree is an interesting open problem, with the understanding that optimal here means subject to the current state of the art.

## 12 Concluding Remarks

We have presented the first large-scale experiment which makes essential recursive use of the elimination step of the GKZ quasi-polynomial algorithm and in doing so have set a new discrete logarithm record, in the field $\mathbb{F}_{2^{30750}}$. We have contributed novel algorithmic and arithmetic

improvements to degree two elimination – the core of the GKZ algorithm – as well as to other small degree eliminations, which together made the computation feasible.

Regarding open problems, the remaining major open problem in fixed characteristic discrete logarithm research is whether or not there is a polynomial time algorithm, either rigorous or heuristic. Since we now have $L(o(1))$ algorithms, one might hope that $L(0)$ complexity is not only possible, but may be discovered in the not too distant future.


## Acknowledgements

The authors would like to thank EPFL's Scientific IT and Application Support (SCITAS) as well as the School of Computer and Communication Sciences (IC) for providing computing resources and support. The first and fourth listed authors were supported by the Swiss National Science Foundation via grant number 200021-156420.


## Appendix A

This Magma script displays the partial factorisation of the group order $N = 2^{30750}-1$ by listing the known prime factors of bit-length up to 135, in increasing order.

```
M := [
   < 1, 1.5850, 3^2 >,
   < 2, 2.8074, 7 >,
   < 3, 3.4594, 11 >,
   < 4, 4.9542, 31 >,
   < 5, 6.3750, 83 >,
   < 6, 7.2384, 151 >,
   < 7, 7.9715, 251 >,
   < 8, 8.3707, 331 >,
   < 9, 9.2312, 601 >,
  < 10, 9.5294, 739 >,
  < 11, 9.5527, 751 >,
  < 12, 10.266, 1231 >,
  < 13, 10.815, 1801 >,
  < 14, 11.136, 2251 >,
  < 15, 11.984, 4051 >,
  < 16, 12.587, 6151 >,
  < 17, 13.706, 13367 >,
  < 18, 15.586, 49201 >,
  < 19, 16.621, 100801 >,
  < 20, 17.335, 165313 >,
  < 21, 21.103, 2252951 >,
  < 22, 21.488, 2940521 >,
  < 23, 21.862, 3813001 >,
  < 24, 21.890, 3887047 >,
  < 25, 21.945, 4036451 >,
  < 26, 23.333, 10567201 >,
  < 27, 27.091, 142958801 >,
  < 28, 27.294, 164511353 >,
  < 29, 27.775, 229668251 >,
  < 30, 31.188, 2446716001 >,
  < 31, 31.342, 2721217151 >,
  < 32, 33.040, 8831418697 >,
  < 33, 36.030, 70171342151 >,
  < 34, 36.276, 83209081801 >,
  < 35, 37.969, 269089806001 >,
  < 36, 39.774, 940217504251 >,
  < 37, 40.044, 1133836730401 >,
  < 38, 42.576, 6554658923851 >,
  < 39, 50.143, 1243595348645401 >,
  < 40, 53.551, 13194317913029593 >,
  < 41, 55.544, 52546634194528801 >,
  < 42, 56.364, 92757531554705041 >,
  < 43, 57.302, 177722253954175633 >,
  < 44, 62.031, 4710883168879506001 >,
  < 45, 70.172, 1330118582061732221401 >,
  < 46, 70.838, 2110663691901109218751 >,
  < 47, 72.225, 5519485418336288303251 >,
  < 48, 74.080, 19963778429046466946251 >,
  < 49, 76.045, 7793957766761995303800 1 >,
  < 50, 81.503, 3427007094604641668368081 >,
  < 51, 81.757, 4086509101824283902341251 >,
  < 52, 93.333, 124775213323021157386615042 01 >,
  < 53, 101.53, 365572506550879718167407895968 1 >,
  < 54, 104.02, 2051873819967980548712143583500 1 >,
  < 55, 112.04, 5346075695594340248521086884817251 >,
  < 56, 114.78, 3575863313159690068505137895414100 1 >,
  < 57, 118.65, 5197244882237713513579066741526383 51 >,
  < 58, 134.30, 2681512326667048810592666965226638 1711401 >
];
```

## Appendix B

The following Magma script verifies the solution of the chosen DLP.

```
F2 := GF(2);
F2T<T> := PolynomialRing(F2);
F2_30<t> := ext< F2 | T^30 + T + 1 >;
F2_30X<X> := PolynomialRing(F2_30);
Xmod := X^1025 + X + t^3;
Fqx<x> := ext<F2_30 | Xmod >;

g := x + t^9;

pi := Pi(RealField(10000));
hpi := &+[ (Floor(pi * 2^(e+1)) mod 2) * t^(29-(e mod 30)) * x^(e div 30): e in [0..30749]];

log := 434390305789220128646032802928291982609103161559747835324845605488213355998660010229578378323680044\
057158457261831399280637186819845679147078090141946891925626461045397701630007678616702582297304813683395 9\
087846766546606415267465500565239163717201376077087517609966272120689864559526335873680408746297753338212 3\
231513130289170500197774868688929331905007130240968186179965256829630759723235184986643193474665087595924 5\
060267190218776740164178112260488483815931431748737437847467299976361467747473017387842849350541942579168 4\
608016179778880005072454767121317594934690955687437370595210441669853744407776185159167679687232354913805 2\
386066567942485370737685601887361218330677074319236862740225949534710215238026301440537483555041065423919 7\
266784536179973336385394753335168947029330190805482576973957270065266397184714930935309467446628613257504 8\
618765736450928587163947994137082514307027043766005401934183541626049464509024096352101110353254477704650 9\
153619133290400508739389603553011299742823890291008572582958577664121160557860000526307205437339364487739 2\
609200421833493193819761982371331256841936976220993608213192351536932044122769920762563332537204366430745 6\
491322128818792893479826432128955657612541674375194974819413295443280251440121265671885606833707185023997 0\
240433143538183503633472532740713186687174872385640926769757644274691898297506228103636556676763923632980 81\
244178174567420620068180279226754002888439307430832622889265703936699056945074221962725618676693802193828 7\
960365690656441673326827531995730974055192335819857917615328755576424290442147211428370989998425531446643 3\
040490438402774574660527270940621439407728182757233942613621964084742108730060269538354222002990932837588 6\
157837711642276338166524972440081627908527487140277227051053118878974916720567207268745925799084333559674 2\
618768667153891226864951915437736871589815193345309879317250408431591179908103523291312706219873671133478 1\
071762692324702166090104421602276440894257176129577107453055178317456720834488950435025810746260853423511 9\
467570686877636607646514699137630041889257122111397640399886286441834427613535770186587442297332515007974 6\
750176848371823737002510621899382555107629024504668087268127214835081579475988791226068329089979591506088 0\
601370394366441950562119422811823176221238788873295319950874683012802994825145631947554377609200917857771 6\
108292333763042356792424304189682026080256757736267162346093543873903555841443418322416494420155604928790 7\
799185130780463361562995275733924670177564706787096974093224634621910071023645348714835033340680626583962 \
718185872073779309256899533909650086388377444012809463896058868285816710932051753889403084055772991089745 2\
905245725814680650447984431265293396393434503978292098131661802134375349226190483358565264051083954048880 4\
261225267023052607121195293840270344868287232506191771013340117976244354450625548520759867492491970954395 6\
048031879032116174083229203065115118597616925653620066479038013914303226388854698172920040114263210311139 2\
406409859925600644800948135491211859484230529549731155281643621460589367483174040434665436664333147558145 7\
082219480928472489283166825413787555460726934830294090663674539192977774881751933512376550897770005120438 3\
753368728878787779315690698598301530065832932507025377206859507455706732825577459360199502674515377920668 6\
965067358666979493492484002240551199481357133786572228651903419875421451051570299360548137501550208630032 5\
601815935582503277216769177013654429675648171060386426822729050673385954432518999466423370391869555679081 1\
131561021002676362618692819221164659444442050944489257491415848532903972386536674618476717500863773048656 \
344678552479111604821239825586796398476060974169145561092164448244920062675358677475293297401716733958203 1\
079023977927594502919824005662773843266285994500235765345174014528478556091136399699120362631954687087170 5\
680802187870021695029815300548641442759664155244154176846066971114559836120505176366449075904116928442959 9\
036586592973239235785713245961452905206099524808907203442943692156650781954380323719287194802414191651873 6\
841890201464197818485501310450022170783877646363464306827794717268130940311514790782451075670789377449421 \
920794893051172026217634173872360689209410517011999427926150174315699456121312625664862760822794722321384 9\
501386189026418912212940151543667714387876504918783153784027425442438243964346915421372149114426906729636 1\
605143275377991353887311363393651857453248698252332298870667791421927040702655665052453856586437106477474 8\
486760200334492982589568705757295368001748207664597750959709165960182325052583533625611559672145587141437 4\
635156005162617927298058431492567011118504223281788046722602668291391102373378486785218964048930277353768 3\
571719690735076554036851088781102160241449797374599544373733833553302702395056738967188186049329085419475 0\
454517733060793738352901583499107322463242388534379818739284602445115889869831003295077745574792380090518 2\
539277945653972077544878139122867814889074242496856284598799558652490205828007996271716258215537083545478 9\
165013317717613215404025318877080086912367686992355463517222456758767158718528085508681093285477482937785 9\
540481631215843659614311506278170888244879602355940924090608094732096788175940519578713906387623435440596 4\
823553558970664968654355621216029992254379848987387103981445305221741917681199188155822304230240090459946 6\
106646917887744792033737774282657499118193369526709149791836408985439218303926239159091783516024020565397 0\
751700892929092904517713865804626853604064142045915107706895854353366744752752985289954855404385661263910 \
516946784126076887628640910324112171259601424394961591498801232330260493524166487366800969054050302258025 3\
009288688508791303914048658359525277275194810825382971012268265684119066232301043490495332092271507544554 4\
695340051218931623315095731325329552278440275154789849198398094684947230417186360010181201828834997230543 4\
367514235998493134875736608458672459178529199349279486228716392267415253025224339624589659467597948329387 6\
988251966205382443483935575706542210807474417922476376339375644210514544291263558851016112487492130521483 4\
```

```
26132975137120072180642757995349329415176886712186646485931813129930338972568153545971126180200474992803 93\
93148312008086318655109840831675595338148423736741052902452764886145577155269963850853522729130995234329 35\
57901864235944465188967907560170396845080626343808937838563495124548040123643688010296637571924605258977 38\
29572215683964873635037217417744499280071130904438645685633457913843139879882812954921626477924357615300 22\
82294774439411003305781279708391536094008259177287116872629622643204406633000444096922118671331921775937 01\
88436838108198130137163037953503369073694427969445801981847021293216215879229713107616037774865104593271 19\
20982393367900411152355866529512344050201717272229205965068100931821730177893518429566572670870877081473998 02\
42895607977118504802429236093184603802241803434651015637762598671763109493187500729224317574299761107742 86\
83190995811656225179905487288565748526303980331098658497418986675554802489023904551593116713668184752537 43\
13897530302845096111331262524374834476920379062195778543138664008471134292533788001863401884222822717267 127\
74469417988000134078850560071285977894946248649152763439000315594216213862869518160407498798636650062236 42\
79683792311555539861454759837193197683424617938567482156209590210168267189814965502609694701477553795010 57\
08710784873196357144061597841238234148566613081351288696419599714541717648306074763143685993092835324595 96\
51623785316560655266717219431670706406788959869931791775229703156148232902524732789029927886323829754916 94\
62594587656281613227793642841626959551873567507196480878309693665093019244782549166026499331971456440726 12\
26008060948741622845755370237259677749115984309906162069954509983886913128260804555932446772278836649392 75\
87431806676687115745660001922677480900680270109892371331746866939642969485802950808464329718361625383291 65\
89565619645864372170206539147691134663896525833568405171212560819056790398718450716087426009375254386833 01\
88718274288146659855389701851650016856122862742320745073204097850046426232888263378856339797499128687068 27\
62984107765683239863840425790840309812803306133842021875540639205664924756191425081845046231039069076727 60\
74241432608387788804632802479265230869544079944138170294679522104857109538350971026011089144102986589082 88\
03964352050774119420218115337082042699706499775767179489088475735766769547067461701901440319138732159387 34\
00972373054512046815343952377032261731759701896615277849573485646582687146148245896100769014957428660426 11\
99058739785062895095691055626962450656994174210495121965594801773965821989133055056237802213561284170759 02\
48901051233242923795643665285676375068867248186172474214474871200212745882459336109077896380962411318356 23\
91861204716944963291161577620373714121905593270544464157873170531802631682017228314586333326591364497256 05\
92840594365610138921475939942404447282207322235477259183394218972871252519365965280479377613145797616536 96\
91421165846428167664146824506022321992824926891403356679803138978440363213380080628132777152423349813889 78\
63456719612286430841641921944366874945836697511776320322923478168678926194446816240626121256736354995202 81\
74039447785587265037383842565952849585667657226261124325298770440030577440402228972107992671962105960303 26\
58773499771320369123207581270215747585620 9;

// If the following is true then the verification was successful
hpi eq g^log;
```

## References


1. Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Advances in Cryptology—EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 1–16. Springer, 2014.
2. Antonia W. Bluher. On $x^{q+1} + ax + b$. *Finite Fields Appl.*, 10(3):285–305, 2004.
3. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997.
4. Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann. 795-bit factoring and discrete logarithms. NMBRTHRY list, 02/12/2019.
5. Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann. Factorization of RSA-250. NMBRTHRY list, 28/02/2020.
6. Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *J. Cryptology*, 31(3):885–916, 2018.
7. Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Inform. Theory*, 30(4):587–593, 1984.
8. Don Coppersmith. Modifications to the number field sieve. *J. Cryptology*, 6(3):169–180, 1993.
9. Andreas Enge and Pierrick Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arith.*, 102:83–103, 2002.
10. Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. 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 *Advances in Cryptology—CRYPTO 2013*, volume 8043 of *LNCS*, pages 109–128. Springer, 2013.
11. Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Solving a 6120-bit DLP on a desktop computer. In *Selected Areas in Cryptography—SAC 2013*, volume 8282 of *LNCS*, pages 136–152. Springer, 2014.
12. Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Discrete Logarithms in $GF(2^{6120})$. NMBRTHRY list, 11/4/2013.
13. Faruk Göloğlu, Robert Granger, Gary McGuire, and Jens Zumbrägel. Discrete Logarithms in $GF(2^{1971})$. NMBRTHRY list, 19/2/2013.
14. Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In *Advances in Cryptology—EUROCRYPT 2016*, pages 263–293. Springer, 2016.

15. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves - (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$). In *Advances in Cryptology—CRYPTO 2014*, volume 8617 of *LNCS*, pages 126–145. Springer, 2014.
16. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the discrete logarithm problem in finite fields of fixed characteristic. *Trans. Amer. Math. Soc.*, 370:3129–3145, 2018.
17. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the powers of 2. Available from `eprint.iacr.org/2014/300`, 29th Apr 2014.
18. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Discrete logarithms in the Jacobian of a genus 2 supersingular curve over $GF(2^{367})$. NMBRTHRY list, 30/1/2014.
19. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Discrete Logarithms in $GF(2^{9234})$. NMBRTHRY list, 31/1/2014.
20. Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. In *Selected Areas in Cryptography—SAC 2013*, volume 8282 of *LNCS*, pages 355–379. Springer, 2014.
21. Antoine Joux. Discrete Logarithms in $GF(2^{1778})$. NMBRTHRY list, 11/2/2013.
22. Antoine Joux. Discrete Logarithms in $GF(2^{6168})$. NMBRTHRY list, 21/5/2013.
23. Antoine Joux. Discrete Logarithms in $GF(2^{4080})$. NMBRTHRY list, 22/3/2013.
24. Antoine Joux and Cécile Pierrot. Improving the polynomial time precomputation of Frobenius representation discrete logarithm algorithms. In *Advances in Cryptology—ASIACRYPT 2014*, pages 378–397. Springer, 2014.
25. Thorsten Kleinjung. Discrete Logarithms in $GF(2^{1279})$. NMBRTHRY list, 17/10/2014.
26. Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen K. Lenstra, Emmanuel Thomé, Joppe W. Bos, Pierrick Gaudry, Alexander Kruppa, Peter L. Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit RSA modulus. In *Advances in Cryptology—CRYPTO 2010*, pages 333–350. Springer, 2010.
27. Thorsten Kleinjung, Joppe W. Bos, and Arjen K. Lenstra. Mersenne factorization factory. In *Advances in Cryptology—ASIACRYPT 2014*, LNCS, pages 358–377. Springer, 2014.
28. Thorsten Kleinjung, Claus Diem, Arjen K. Lenstra, Christine Priplata, and Colin Stahlke. Computation of a 768-bit prime field discrete logarithm. In *Advances In Cryptology—EUROCRYPT 2017, Pt I*, volume 10210 of *LNCS*, pages 185–201. Springer, 2017.
29. Thorsten Kleinjung and Benjamin Wesolowski. Discrete logarithms in quasi-polynomial time in finite fields of fixed characteristic. Available from `eprint.iacr.org/2019/751`, 25th June 2019.
30. Brian A. LaMacchia and Andrew M. Odlyzko. Solving large sparse linear systems over finite fields. In *Advances in Cryptology—CRYPTO '90*, volume 537 of *LNCS*, pages 109–133. Springer, 1991.
31. Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Research Nat. Bur. Standards*, 45:255–282, 1950.
32. Hendrik W. Lenstra, Jr. Finding isomorphisms between finite fields. *Math. Comp.*, 56:329–347, 1991.