MATHEMATICS OF COMPUTATION
Volume 88, Number 317, May 2019, Pages 1391–1426
https://doi.org/10.1090/mcom/3363
Article electronically published on June 19, 2018

COMPUTING ISOMORPHISMS AND EMBEDDINGS OF FINITE FIELDS

LUDOVIC BRIEULLE, LUCA DE FEO, JAVAD DOLISKANI,
JEAN-PIERRE FLORI, AND ÉRIC SCHOST

ABSTRACT. Let $\mathbb{F}_q$ be a finite field. Given two irreducible polynomials f, g over $\mathbb{F}_q$, with $\deg f$ dividing $\deg g$, the finite field embedding problem asks to compute an explicit description of a field embedding of $\mathbb{F}_q[X]/f(X)$ into $\mathbb{F}_q[Y]/g(Y)$. When $\deg f = \deg g$, this is also known as the isomorphism problem.

This problem, a special instance of polynomial factorization, plays a central role in computer algebra software. We review previous algorithms, due to Lenstra, Allombert, Rains, and Narayanan, and propose improvements and generalizations. Our detailed complexity analysis shows that our newly proposed variants are at least as efficient as previously known algorithms, and in many cases significantly better.

We also implement most of the presented algorithms, compare them with the state of the art computer algebra software, and make the code available as an open source. Our experiments show that our new variants consistently outperform available software.

1. Introduction

Let q be a prime power and let $\mathbb{F}_q$ be a field with q elements. Let f and g be irreducible polynomials over $\mathbb{F}_q$, with $\deg f$ dividing $\deg g$. Define $k = \mathbb{F}_q[X]/f(X)$ and $K = \mathbb{F}_q[Y]/g(Y)$; then, there is an embedding $\phi : k \hookrightarrow K$, unique up to $\mathbb{F}_q$-automorphisms of k. The goal of this paper is to describe algorithms to efficiently represent and evaluate one such embedding.

All the algorithms we are aware of split the embedding problem into two subproblems:

(1) Determine elements $\alpha \in k$ and $\beta \in K$ such that $k = \mathbb{F}_q(\alpha)$, and such that there exists an embedding ϕ mapping α to β. We refer to this problem as the *embedding description problem*. It is easily seen that (α, β) describes an embedding if and only if α and β share the same minimal polynomial.

(2) Given elements α and β as above, given $\gamma \in k$ and $\delta \in K$, solve the following problems:
- Compute $\phi(\gamma) \in K$.
- Test if $\delta \in \phi(k)$.
- If $\delta \in \phi(k)$, compute $\phi^{-1}(\delta) \in k$.

We refer collectively to these problems as the *embedding evaluation problem*.

Received by the editor May 5, 2017, and, in revised form, July 3, 2017, and December 18, 2017.

2010 *Mathematics Subject Classification.* Primary 13P05, 68W30.

Motivation, previous work. The first to become interested in this problem was H. Lenstra. In his seminal paper [36] he shows that it can be solved in deterministic polynomial time, by using a representation for finite fields that he calls *explicit data*.[1] In practice, the embedding problem arises naturally when designing a computer algebra system: as soon as a system is capable of representing arbitrary finite fields, it is natural to ask it to compute the morphisms between them. Ultimately, by representing effectively the lattice of finite fields with inclusions, the user is given access to the algebraic closure of $\mathbb{F}_q$. The first system to implement a general embedding algorithm was Magma [4]. As detailed by its developers [5], it used a much simpler approach than Lenstra's algorithm, entirely based on polynomial factorization and linear algebra. Lenstra's algorithm was later revived by Allombert [2, 3] who modified some key steps in order to make it practical; his implementation has since been part of the PARI/GP system [54].

Meanwhile, a distinct family of algorithms for the embedding problem was started by Pinch [46], and later improved by Rains [48]. These algorithms, based on principles radically different from Lenstra's, are intrinsically probabilistic. Although their worst-case complexity is no better than that of Allombert's algorithm, they are potentially much more efficient on a large set of parameters. This potential was understood by Magma's developers, who implemented Rains' algorithm in Magma v2.14.[2]

With the exception of Lenstra's work, the aforementioned papers were mostly concerned with the practical aspects of the embedding problem. While it was generally understood that computing embeddings is an easier problem than general polynomial factoring, no results on its complexity more precise than Lenstra's had appeared until recently. A few months before the present paper was finalized, Narayanan published a novel generalization of Allombert's algorithm [42], based on elliptic curve computations, and showed that its (randomized) complexity is at most quadratic. Narayanan's generalization relies on the fact that Artin–Schreier and Kummer theories are special cases of a more general situation: as already emphasized by Couveignes and Lercier [11], whereas the former theory acts on the additive group of a finite field, and the latter on its multiplicative group, they can be extended to more general commutative algebraic groups, in particular, to elliptic curves.

Our contribution. This work aims to be, in large part, a complete review of all known algorithms for the embedding problem; we analyze in detail the cost of existing algorithms and introduce several new variants. After laying out the foundations in the next section, we start with algorithms for the embedding description problem.

Section 3 describes the family of algorithms based (more or less loosely) on Lenstra's work; we call these *Kummer-type* algorithms. In doing so, we pay a particular attention to Allombert's algorithm: to our knowledge, this is the first detailed and complete complexity analysis of this algorithm and its variants. Thanks to our work on asymptotic complexity, we were able to devise improvements to the original variants of Allombert that largely outperform them both in theory and

[1] Technically, Lenstra only proved his theorem in the case where k and K are isomorphic; however, the generalization to the embedding problem poses no difficulties.

[2] As a matter of fact, Rains' algorithm was never published; the only publicly available source for it is in Magma's source code (file `package/Ring/FldFin/embed.m`, since v2.14).

practice. One notable omission in this section is Narayanan's algorithm, which is, in our opinion, mostly of theoretical rather than practical interest. We present instead in Subsection 3.3 a simpler algorithm with essentially the same complexity.

In Section 4 we describe Rains' algorithm. Rains' original preprint [48] went unpublished, thus we give here a complete description and analysis of his algorithm, for reference. We also give new variants of Rains' algorithm of lesser interest in Appendix A.

Then, in Section 5 we present a generalization of Rains' algorithm using elliptic curves. The possibility of this algorithm was hinted at by Rains, but never fully developed; we show that it is indeed possible to use *elliptic periods* to solve the embedding description problem, and that the resulting algorithm behaves well both in theory and in practice. While working out the correctness proof of the elliptic variant of Rains' algorithm, we encounter an unexpected difficulty: whereas roots of unity enjoy Galois properties that guarantee the success of Rains' original algorithm, points of elliptic curves fail to provide the same. Heuristically, the failure probability of the elliptic variant is extremely small, however, we are not able to prove it formally. Our experimental searches even seem to suggest that the failure probability might be, surprisingly, zero. We state this as a conjecture on elliptic periods (see Conjecture 23).

Section 6 does a global comparison of all the algorithms presented previously. In particular, Rains' algorithm and variants require a non-trivial search for parameters, which we discuss thoroughly. Then we present an algorithm to select the best performing embedding description algorithm from a practical point of view. This theoretical study is complemented by the experimental Section 7, where we compare our implementations of all the algorithms; our source code is made available through the Git repository `https://github.com/defeo/ffisom` for replication and further scrutiny.

The algorithms for the embedding evaluation problem are much more classical and well understood. Due to space constraints, we do not present them here; we address instead the interested reader to the extended version of the present paper [7].

In conclusion, we hope that our review will constitute a reference guide for researchers and engineers interested in implementing embeddings of finite fields in a computer algebra system.

2. Preliminaries

2.1. **Fundamental algorithms and complexity.** We review the fundamental building blocks that constitute the algorithms presented next. We are going to measure all complexities in number of operations $+$, $\times$, $\div$ in $\mathbb{F}_q$, unless explicitly stated otherwise. Most of the algorithms we present are randomized; we use the *big-Oh* notation $O(\,)$ to express *average* asymptotic complexity, and we will make it clear when this complexity depends on heuristics. We also occasionally use the notation $\tilde{O}(\,)$ to neglect logarithmic factors in the parameters.

We let $\mathsf{M}(m)$ be a function such that polynomials in $\mathbb{F}_q[X]$ of degree less than m can be multiplied in $\mathsf{M}(m)$ operations in $\mathbb{F}_q$, under the assumptions of [55, Ch. 8.3], together with the slightly stronger one, that $\mathsf{M}(mn)$ is in $O(m^{1+\varepsilon}\mathsf{M}(n))$ for all $\varepsilon > 0$. Using FFT multiplication, one can take $\mathsf{M}(m) \in O(m \log(m) \operatorname{loglog}(m))$ [8].

We denote by ω the *exponent of linear algebra*, i.e., a constant such that $m \times m$ matrices with coefficients in any field k can be multiplied using $O(m^\omega)$ additions and multiplications in k. One can take $\omega < 2.38$, the best result to date being in [34]; on the other hand, we also suppose that $\omega > 2$.

The algorithms presented in the next sections perform computations in ring extensions of finite fields. Some of these extensions also happen to be finite fields. As customary, if k is a finite field and ξ is some element of an algebraic extension of k, we will write $k[\xi]$ for the ring generated by ξ. To avoid confusion, when the extension generated by ξ is a finite field, we will write instead $k(\xi)$.

Some algorithms will operate in a polynomial ring $k[Z]$, where k is a field extension of $\mathbb{F}_q$; some other algorithms will operate in $k[Z]/h(Z)$, where h is a monic polynomial in $k[Z]$. We review the basic operations in these rings. We assume that k is represented as a quotient ring $\mathbb{F}_q[X]/f(X)$, with $m = \deg f$, and we let $s = \deg h$ in the complexity estimates.

Multiplying and dividing polynomials of degree at most s in $k[Z]$ is done in $O(\mathsf{M}(sm))$ operations in $\mathbb{F}_q$, using Kronecker's substitution [20, 22, 39, 55, 56]. Multiplication in $k[Z]/h(Z)$ is also done in $O(\mathsf{M}(sm))$ operations using the technique in [44]. By the same techniques, gcds of degree m polynomials in $k[Z]$ and inverses in $k[Z]/h(Z)$ are computed in $O(\mathsf{M}(sm)\log(sm))$ operations.

Given polynomials $e, g, h \in k[Z]$ of degree at most s, modular composition is the problem of computing $e(g) \bmod h$. An upper bound on the algebraic complexity of modular composition is obtained by the Brent–Kung algorithm [6]; under our assumptions on the respective costs of polynomial and matrix multiplication, its cost is $O(s^{(\omega+1)/2}\mathsf{M}(m))$ operations in $\mathbb{F}_q$ (so if $k = \mathbb{F}_q$, this is $O(s^{(\omega+1)/2})$). In the binary RAM complexity model, the Kedlaya–Umans algorithm [25] and its extension in [47] yield an algorithm with essentially linear complexity in s, m and $\log(q)$. Unfortunately, making these algorithms competitive in practice is challenging; we are not aware of any implementation of them that would outperform Brent and Kung's algorithm.

Note 1. If we have several modular compositions of the form $e_1(g) \bmod h, \dots, e_t(g) \bmod h$ to compute, we can slightly improve the obvious bound $O(ts^{(\omega+1)/2})$ (we discuss here $k = \mathbb{F}_q$, so $m = 1$). If $t = O(s)$, using [24, Lemma 4], this can be done in time $O(t^{(\omega-1)/2}s^{(\omega+1)/2})$. If $t = \Omega(s)$, this can be done in $O(ts^{\omega-1})$ operations, by computing $1, g, \dots, g^{s-1}$ modulo f, and doing a matrix product in size $s \times s$ by $s \times t$.

Frobenius evaluation. Consider an $\mathbb{F}_q$-algebra Q, and an element α in Q. Given integers c, d, we will have to compute expressions of the form

$$\sigma_d = \alpha^{q^d}, \quad \tau_d = \sum_{i=0}^{d-1} \alpha^{q^{ci}}, \quad \mu_d = \alpha^{\lfloor q^d/c \rfloor} \ .$$

A direct binary powering approach would yield a complexity of, e.g., $O(d\log(q))$ multiplications in Q for the first expression.

To do better, we use a recursive approach that goes back to [56], with further ideas borrowed from [23, 52]. For $i \geq 1$, define integers A_i, B_i as follows:

$$q^i = A_i c + B_i, \quad 0 \leq B_i < c.$$

Then, we have the relations

$$\sigma_{i+j} = \sigma_j^{q^i}, \quad \tau_{i+j} = \tau_i + \tau_j^{q^{ic}}, \quad \mu_{i+j} = \mu_j^{q^i} \mu_i^{B_j} \alpha^{\lfloor B_i B_j / c \rfloor}.$$

Since we are interested in σ_d, τ_d and μ_d, using an addition chain for d, we are left to perform $O(\log(d))$ steps as above.

To perform these operations, we will make heavy use of a technique originating in [56]. In its simplest form, it amounts to the following: if $Q = \mathbb{F}_q[X]/f(X)$, for some polynomial f in $\mathbb{F}_q[X]$, and β is in Q, we can compute β^q by means of the modular composition $\beta(\xi)$, where $\xi = x^q$ and x is the image of X modulo f.

In the following proposition, we discuss versions of this idea for various kinds of algebras Q, and how they allow us to compute the expressions σ_d, τ_d, μ_d defined above.

Proposition 2. *Let $f \in \mathbb{F}_q[X]$ be a polynomial of degree m, and define the $\mathbb{F}_q$-algebra $Q = \mathbb{F}_q[X]/f(X)$. Let $h \in Q[Z]$ be a polynomial of degree s, and define the Q-algebra $S = Q[Z]/h(Z)$. Finally, whenever $h \in \mathbb{F}_q[Z]$, define the $\mathbb{F}_q$-algebra $Q' = \mathbb{F}_q[Z]/h(Z)$.*

Denote by $T_Q, T_S, T_{Q'}$ the cost, in terms of $\mathbb{F}_q$-operations, of one modular composition in Q, S, Q', respectively. Also, denote by $T_{Q,t} \leq tT_Q$ (resp. $T_{S,t}, T_{Q',t}$) the cost of t modular compositions sharing the same polynomial (see Note 1).

Then the expressions

$$\sigma_d = \alpha^{q^d}, \quad \tau_d = \sum_{i=0}^{d-1} \alpha^{q^{ci}}, \quad \mu_d = \alpha^{\lfloor q^d / c \rfloor}$$

can be computed using the following number of operations:

Case 1. *$\alpha \in Q$:*
- *σ_d: $O(\mathsf{M}(m)\log(q) + T_Q \log(d))$,*
- *τ_d: $O(\mathsf{M}(m)\log(q) + T_Q \log(dc))$,*
- *μ_d: $O(\mathsf{M}(m)\log(q) + (T_Q + \mathsf{M}(m)\log(c))\log(d))$;*

Case 2. *$\alpha \in Q$ with $f | X^r - 1$:*
- *σ_d: $O(\mathsf{M}(m)\log(q) + \mathsf{M}(r)\log(d))$,*
- *τ_d: $O(\mathsf{M}(m)\log(q) + \mathsf{M}(r)\log(dc))$,*
- *μ_d: $O(\mathsf{M}(m)\log(q) + (\mathsf{M}(r) + \mathsf{M}(m)\log(c))\log(d))$;*

Case 3. *$\alpha \in S$:*
- *σ_d: $O(\mathsf{M}(ms)\log(q) + (T_{Q,s} + T_S)\log(d))$,*
- *τ_d: $O(\mathsf{M}(ms)\log(q) + (T_{Q,s} + T_S)\log(dc))$,*
- *μ_d: $O(\mathsf{M}(ms)\log(q) + (T_{Q,s} + T_S + \mathsf{M}(ms)\log(c))\log(d))$;*

Case 4. *$\alpha \in S$ with $h \in \mathbb{F}_q[Z]$:*
- *σ_d: $O((\mathsf{M}(m) + \mathsf{M}(s))\log(q) + (T_{Q,s} + T_{Q',m})\log(d)$,*
- *τ_d: $O((\mathsf{M}(m) + \mathsf{M}(s))\log(q) + (T_{Q,s} + T_{Q',m})\log(dc))$,*
- *μ_d: $O((\mathsf{M}(m) + \mathsf{M}(s))\log(q) + (T_{Q,s} + T_{Q',m} + (m\mathsf{M}(s) + s\mathsf{M}(m))\log(c))\log(d))$;*

Case 5. *$\alpha \in S$ with $h | X^r - a$ for $a \in Q$:*
- *σ_d: $O(\mathsf{M}(m)\log(q) + (T_{Q,s} + \mathsf{M}(mr))\log(d))$,*
- *τ_d: $O(\mathsf{M}(m)\log(q) + (T_{Q,s} + \mathsf{M}(mr))\log(dc))$,*
- *μ_d: $O(\mathsf{M}(m)\log(q) + (T_{Q,s} + \mathsf{M}(mr) + \mathsf{M}(m)\log(c))\log(d))$.*

Proof. The complexity estimates mostly rely on the complexity of modular composition.

Case 1. We let x be the image of X in Q, and we start by computing x^q, using $O(\mathsf{M}(m)\log(q))$ operations in $\mathbb{F}_q$.

For $i \geq 0$, given $\xi_i = x^{q^i}$ and β in Q, we can compute β^{q^i} as $\beta^{q^i} = \beta(\xi_i)$, using $T_Q = O(m^{(\omega+1)/2})$ operations; in particular, this allows us to compute ξ_{i+j} from the knowledge of ξ_i and ξ_j. Given an addition chain for d, we thus compute all corresponding ξ_i's, and we deduce the σ_i's similarly, since $\sigma_{i+j} = \sigma_j(\xi_j)$. Altogether, starting from $\xi_1 = x^q$, this gives us σ_d for $O(T_Q \log(d))$ further operations in $\mathbb{F}_q$.

The same holds for τ_d, with a cost in $O(T_Q \log(cd))$, since we have to compute ξ_c first; and for μ_d, with a cost in $O((T_Q + \mathsf{M}(m)\log(c))\log(d))$ operations, as the formula for μ_{i+j} shows that we can obtain it by means of a modular composition (to compute $\mu_j^{q^i} = \mu_j(\xi_i)$), together with two exponentiations of indices less than c.

The costs for computing σ_d, τ_d, μ_d follow immediately.

Case 2. When f divides $X^r - 1$, we obtain $\beta(\xi_i)$ by computing $\beta(X^{q^i \bmod r}) \bmod (X^r - 1)$ first, and then reducing modulo $f(X)$. Thus, the cost of one modular composition is $T_Q = O(\mathsf{M}(r))$, and the total cost is obtained by replacing this value in the estimates for the previous case.

Case 3. We let x and z be the respective images of X in Q and Z in S, and as a first step, we compute z^q (and x^q, unless f is as in Case 2 above), in $O(\mathsf{M}(ms)\log(q))$ operations.

In order to compute the quantities σ_d, τ_d, μ_d, we apply the same strategy as above; the key factor for complexity is thus the cost of computing β^{q^i}, for β in S, given $\zeta_i = z^{q^i}$ and $\xi_i = x^{q^i}$ (as we did in Case 1, we apply this procedure to our input element α, as well as to ζ_i itself, and ξ_i, in order to be able to continue the calculation).

To do so, we use an algorithm by Kaltofen and Shoup [23], which boils down to writing $\beta = \sum_{j=0}^{s-1} c_j(x) z^j$, so that $\beta^{q^i} = \sum_{j=0}^{s-1} c_j(x)^{q^i} \zeta_i^j$. The s coefficients $c_j(x)^{q^i}$ are computed by applying the previous algorithms in Q to s inputs. This takes time at most sT_Q, but as pointed out in Note 1, improvements are possible if we base our algorithm on modular composition; we thus denote the cost $T_{Q,s}$.

Then, we do a modular composition in S to evaluate the result at ζ_i; this latter step takes $T_S = O(s^{(\omega+1)/2}\mathsf{M}(m))$ operations in $\mathbb{F}_q$.

Case 4. The cost for computing z^q is $O(\mathsf{M}(s)\log(q))$ and that for computing x^q is $O(\mathsf{M}(m)\log(q))$. In the last step, the cost T_S of modular composition in S is now that of m modular compositions in degree s (with the same argument), as detailed in Note 1, that we denote $T_{Q',m}$. Similarly, the cost of multiplication in S can be reduced from $O(\mathsf{M}(ms))$ to $O(s\mathsf{M}(m) + m\mathsf{M}(s))$ operations.

Case 5. We start by computing x^q, using $O(\mathsf{M}(m)\log(q))$ operations in $\mathbb{F}_q$.

For β as above, suppose that we have already computed all coefficients $d_j(x) = c_j(x)^{q^i}$ in $O(T_{Q,s})$ operations; we now have to compute $\beta^{q^i} = \sum_{j=0}^{s-1} d_j(x)\zeta_i^j$.

We first do the calculation modulo $Z^r - a$ rather than modulo h; that is, we compute $\sum_{j=0}^{s-1} d_j(x) z_i^j$, where $z_i = z^{q^i}$. Since $z^r = a$, we have $z_i = a_i z^{q^i \bmod r}$, with $a_i = a^{\lfloor q^i/r \rfloor}$. If we assume that a_i is known, we can compute $\sum_{j=0}^{s-1} d_j(x) z_i^j$ using Horner's method, in time $O(s\mathsf{M}(m))$, and we reduce this result modulo h, for the cost $O(\mathsf{M}(mr))$ of a Euclidean division in degree r in $Q[Z]$.

In order to continue the calculation for all indices in our addition chain, we must thus compute the corresponding a_i's as well, just like the μ_i's; this takes $O(T_Q + \mathsf{M}(m)\log(r))$ operations.

Since the first stage of the algorithm took $O(T_{Q,s})$ operations, we can take $T_S = O(\mathsf{M}(mr))$ for computing β^{q^i}.

To initiate the procedure, the algorithm also needs to compute $a_1 = a^{\lfloor q/r \rfloor}$, using $O(\log(q))$ multiplications in Q for a cost $O(\mathsf{M}(m)\log(q))$. □

Computing subfields. With $k = \mathbb{F}_q[X]/f(X)$ and $\deg f = m$ as above, we are given a divisor r of m, and we want to construct an intermediate extension $\mathbb{F}_q \subset L \subset k$ of degree r over $\mathbb{F}_q$. More precisely, we want to compute a monic irreducible polynomial $g \in \mathbb{F}_q[X]$ of degree r, and a polynomial $h \in \mathbb{F}_q[X]$ such that $x \mapsto h(x) \bmod f$ defines an embedding $L = \mathbb{F}_q[X]/g(X) \hookrightarrow k$.

An algorithm for this will be used to reduce the embedding problem to an isomorphism problem; although this is not strictly necessary for most of our algorithms to work, it will greatly simplify the expostion. We proceed as follows.

Let $\alpha \in k$ be a random element[3]. Then α has a minimal polynomial of degree m over $\mathbb{F}_q$ with high probability. In other words, one needs $O(1)$ such random elements to find one with degree m minimal polynomial. Now, the trace

$$\mathrm{Tr}_{k/L}(\alpha) = \alpha + \alpha^{q^r} + \cdots + \alpha^{q^{m-r}} \tag{1}$$

has a minimal polynomial of degree r over $\mathbb{F}_q$ with high probability as well. This means we can compute, after $O(1)$ random trials, the desired polynomials $\beta = \mathrm{Tr}_{k/L}(\alpha)$, its minimal polynomial g, and h the polynomial of degree less than m representing β.

Proposition 3. *Let $\mathbb{F}_q \subset k$ be a finite extension of degree m, and let r be a divisor of m. Computing an intermediate field $\mathbb{F}_q \subset L \subset k$ with $[L : \mathbb{F}_q] = r$ takes an expected $O(m^{(\omega+1)/2}\log(m) + \mathsf{M}(m)\log(q))$ operations in $\mathbb{F}_q$. Once L is computed, any element $\gamma \in L$ can be lifted to its image in k using $O(m^{(\omega+1)/2})$ operations.*

Proof. Computing the minimal polynomial of an element in k takes $O(m^{(\omega+1)/2})$ operations in $\mathbb{F}_q$; see [53]. The trace in (1) is computed as the expression τ_m of the previous paragraph (with $c = r$ and $d = m/r$), at a cost of $O(m^{(\omega+1)/2}\log(m) + \mathsf{M}(m)\log(q))$ operations in $\mathbb{F}_q$.

Finally, given an element $\gamma \in L$, its image in k is computed by evaluating $h(\gamma)$, where h is the polynomial representation of $\mathrm{Tr}_{k/L}(\alpha)$. This can be done by a modular composition at cost $O(m^{(\omega+1)/2})$. □

Root finding in cyclotomic extensions. Given a field $k = \mathbb{F}_q[X]/f(X)$ of degree m as above, we will need to factor some special polynomials in $k[Z]$: we are interested in finding one factor of a polynomial that splits into factors of the same, known, degree. This problem is known as *equal degree factorization* (EDF), and the best generic algorithm for it is the Cantor–Zassenhaus method [9, 56], which

[3] For efficiency reasons, it may be preferable to choose α pseudo-randomly, such as a low-degree polynomial in the generator of k.

runs in $O(\mathsf{M}(sm)(dm\log(q)+\log(sm)))$ operations in $\mathbb{F}_q$ [55, Th. 14.9], where s is the degree of the polynomial to factor, and d is the degree of the factors.

More efficient variants of the Cantor–Zassenhaus method are known for special cases. When the degree s of the polynomial is small compared to the extension degree m, Kaltofen and Shoup [23] give an efficient algorithm which is as follows.

Algorithm 1. Kaltofen–Shoup EDF for extension fields

```
Input: A polynomial $h$ with irreducible factors of degree $d$ over $k = \mathbb{F}_q[X]/f(X)$.
Output: An irreducible factor of $h$ over $k$.
1. If $\deg h = d$ return $h$.
2. Take a random polynomial $a_0 \in k[Z]$ of degree less than $\deg h$,
3. Compute $a_1 \leftarrow \sum_{i=0}^{md-1} a_0^{q^i} \mod h$,
4. if $q$ is an even power $q = 2^e$ then
5.     Compute $a_2 \leftarrow \sum_{i=0}^{e-1} a_1^{2^i} \mod h$
6. else
7.     Compute $a_2 \leftarrow a_1^{(q-1)/2} \mod h$
8. end if
9. Compute $h_0 \leftarrow \gcd(a_2, h)$ and $h_1 \leftarrow \gcd(a_2 - 1, h)$ and $h_{-1} \leftarrow h/(h_0 h_1)$,
10. Apply recursively to the smallest non-constant polynomial among $h_0, h_1, h_{-1}$.
```

We refer the reader to the original paper [23] for the correctness of the Kaltofen–Shoup algorithm. We are mainly interested here in its application to root extraction in cyclotomic extensions. Let r be a prime power and let f be an irreducible factor of the rth cyclotomic polynomial Φ_r, with $s = \deg f$. Denote $\mathbb{F}_q[X]/f(X)$ by $\mathbb{F}_q(\zeta)$, where ζ is the image of X in the quotient. Given an rth power $\alpha \in \mathbb{F}_q(\zeta)$ we want to compute an rth root $\alpha^{1/r}$, or equivalently a linear factor of $Z^r - \alpha$ over $\mathbb{F}_q(\zeta)$.

We propose two different algorithms: one of them is quadratic in r, whereas the other one has a runtime that depends on r and s, and will perform better for small values of s.

Proposition 4. *Let r be a prime power and let ζ be a primitive rth root of unity; also let $s = [\mathbb{F}_q(\zeta) : \mathbb{F}_q]$. One can take rth roots in $\mathbb{F}_q(\zeta)$ using either*

$$O(\mathsf{M}(s)\log(q) + rs^{\omega-1}\log(r)\log(s) + \mathsf{M}(rs)\log(s)\log(r))$$

or

$$O(\mathsf{M}(s)\log(q) + r\mathsf{M}(r)\log(s) + \mathsf{M}(rs)\log(s)\log(r))$$

operations in $\mathbb{F}_q$.

Proof. We use Algorithm 1 with $k = \mathbb{F}_q(\zeta)$, to get a linear factor of the polynomial $Z^r - \alpha$, so that $d = 1$ (note that $Z^r - \alpha$ splits into linear factors in $k[Z]$). We discuss Step 3, which is the dominant step. Let $f \in \mathbb{F}_q[X]$ be the defining polynomial of $\mathbb{F}_q(\zeta)$ and let h be a factor of $Z^r - \alpha$ of degree n.

We are in Case 5 of our discussion on Frobenius evaluation, and we want to compute a trace-like expression of the form τ_s. As per that discussion, two algorithms are available to do Frobenius evaluation in k (one of them uses modular composition, the other the fact that f divides $X^r - 1$). Because $s \leq r$, we deduce that a_1 can be computed in either

$$O(\mathsf{M}(s)\log(q) + rs^{\omega-1}\log(s) + \mathsf{M}(rs)\log(s))$$

or

$$O(\mathsf{M}(s)\log(q) + n\mathsf{M}(r)\log(s) + \mathsf{M}(rs)\log(s))$$

operations in $\mathbb{F}_q$, where the first term accounts for computing $\alpha^{\lfloor q/r \rfloor}$ (so we need only compute it once).

Steps 7 and 5 are again an instance of Case 5, and their cost is subsumed by Step 3. The depth of the recursion in Algorithm 1 is $\log(r)$, and the degree n is halved each time, so we obtain the desired result. □

Root finding in some extensions of cyclotomic extensions. Let $r = v^d$, where $v \neq p$ is a prime and d is a positive integer and let s be the order of q in $\mathbb{Z}/v\mathbb{Z}$. We assume that $d \geq 2$, since this will be the case whenever we want to apply the following.

Consider an extension $\mathbb{F}_q \subset k = \mathbb{F}_q[X]/f(X)$ of degree r, and let $\mathbb{F}_q(\zeta)$ and $k(\zeta)$ be extensions of degree s over $\mathbb{F}_q$ and k, respectively, defined by an irreducible factor of the vth cyclotomic polynomial over $\mathbb{F}_q$. In this paragraph, we discuss the cost of computing a vth root in $k(\zeta)$, by adapting the root extraction algorithm given in [16].

Following [16, Algorithm 3], one reduces the root extraction in $k(\zeta)$ to a root extraction in $\mathbb{F}_q(\zeta)$; note that [16, Algorithm 3] reduces the root extraction to the smallest possible extension of $\mathbb{F}_p$, but projecting to $\mathbb{F}_q(\zeta)$ is more convenient here. The critical computation in this algorithm is a trace-like computation performing the reduction.

Algorithm 2. vth root in $k(\zeta)$

```
Input: $a \in k(\zeta)^v$
Output: a $v$th root of $a$
repeat
    choose a random $c \in k(\zeta)$
    $a' \leftarrow ac^v$
    $\lambda \leftarrow a'^{(q^s-1)/v}$
    $b \leftarrow 1 + \lambda + \lambda^{1+q^s} + \cdots + \lambda^{1+q^s+\cdots+q^{(r-2)s}}$
until $b \neq 0$
$\beta \leftarrow (a'b^v)^{1/v}$ in $\mathbb{F}_q(\zeta)$
return $\beta b^{-1} c^{-1}$
```

One multiplication in $k(\zeta)$ amounts to doing r multiplications modulo a degree s factor of Φ_v, and s multiplications modulo f; since $s \leq r$, this takes $O(s\mathsf{M}(r))$ operations in $\mathbb{F}_q$. The computation of $\lambda = a'^{(q^s-1)/v} = a'^{\lfloor q^s/v \rfloor}$ can then be done as explained in our discussion on Frobenius evaluation (Case 4). The cost of each modular composition is $O(s^{(\omega-1)/2} r^{(\omega+1)/2})$, for a total of $O(s^{(\omega-1)/2} r^{(\omega+1)/2} \log(s) + s\mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$.

The trace-like computation of $1 + \lambda + \lambda^{1+q^s} + \cdots + \lambda^{1+q^s+\cdots+q^{(r-2)s}}$ can be done as follows. Let x be the image of X in $k = \mathbb{F}_q[X]/f(X)$. To compute x^{q^s} we first compute x^q using $O(\mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$, and then do $\log(s)$ modular compositions in k. To compute λ^{q^s}, note that an element $\lambda \in k(\zeta)$ can be written as $\lambda = \lambda_0(x) + \lambda_1(x)\zeta + \cdots + \lambda_{s-1}(x)\zeta^{s-1}$ and that $\zeta^{q^s} = \zeta$. Therefore for any i,

$$\lambda^{q^{is}} = \sum_{j=0}^{s-1} \lambda_j(x^{q^{is}}) \left(\zeta^{q^{is}}\right)^j = \sum_{j=0}^{s-1} \lambda_j(x^{q^{is}}) \zeta^j.$$

In particular, given $x^{q^{is}}$, $\lambda^{q^{is}}$ can be computed using $O(s^{(\omega-1)/2}r^{(\omega+1)/2})$ operations in $\mathbb{F}_q$, and [16, Algorithm 2] can be applied in a direct way, with a cost of $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r) + \mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$.

The root extraction in $\mathbb{F}_q(\zeta)$ is done as in the previous paragraph, and have a negligible cost, since we assumed that $s \le v \le \sqrt{r}$. Therefore, we arrive at the following result.

Proposition 5. *With k, ζ and v as above, one can extract vth roots in $k(\zeta)$ using an expected $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r) + s\mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$.*

2.2. **The Embedding Description problem.** We are finally ready to address the problem of describing the embedding of $k = \mathbb{F}_q[X]/f(X)$ in $K = \mathbb{F}_q[Y]/g(Y)$; throughout the paper we let $m = \deg f$ and $n = \deg g$, so that $m|n$. The *embedding description problem* asks to find two elements $\alpha \in k$ and $\beta \in K$ such that $\phi(\alpha) = \beta$ for some field embedding $\phi : k \to K$. This is equivalent to saying that α and β have the same minimal polynomial.

The most obvious way to solve this problem is to take the class of X in $k = \mathbb{F}_q[X]/f(X)$ for α, and a root of f in K for β. Since f splits completely in K, we can apply Algorithm 1 for the special case $d = 1$. Using our discussion on the cost of Frobenius evaluation (precisely, Case 4), we obtain an upper bound of $O\big((nm^{(\omega+1)/2} + \mathsf{M}(m)n^{(\omega+1)/2} + m\mathsf{M}(n)\log(q))\log(m)\big)$ expected operations in $\mathbb{F}_q$ for the problem. We remark that this complexity is strictly larger than $\tilde{O}(m^2)$.

For a more specialized approach, we note that it is enough to solve the following problem: let r be a prime power such that $r|m$ and $\gcd(r, m/r) = 1$, find $\alpha_r \in k$ and $\beta_r \in K$ such that α_r and β_r have the same minimal polynomial, of degree r.

Indeed, once such α_r and β_r are known for every primary factor r of m, possible solutions to the embedding problem are

$$\alpha = \prod_{\substack{r|m,\\ \gcd(r,m/r)=1}} \alpha_r, \qquad \beta = \prod_{\substack{r|m,\\ \gcd(r,m/r)=1}} \beta_r$$

or

$$\alpha = \sum_{\substack{r|m,\\ \gcd(r,m/r)=1}} \alpha_r, \qquad \beta = \sum_{\substack{r|m,\\ \gcd(r,m/r)=1}} \beta_r.$$

Moreover, to treat the general embedding description problem, it is sufficient to treat the case where $[k : \mathbb{F}_q] = [K : \mathbb{F}_q] = r$. Indeed, we can reduce to this situation by applying Proposition 3, at an additional cost of $O(n^{(\omega+1)/2}\log(n) + \mathsf{M}(n)\log(q))$ for each primary factor r. Therefore, to simplify the exposition, we focus on algorithms solving the following problem.

Problem. Let r be a prime power and k, K two extensions of $\mathbb{F}_q$ of degree r. Describe an isomorphism between k and K.

All algorithms presented next are going to rely on one common principle: construct an element in k (and in K) such that its minimal polynomial (or, equivalently, its orbit under the absolute Galois group of $\mathbb{F}_q$) is uniquely (or *almost* uniquely) defined.

3. Kummer-type algorithms

In this section, we review what we call *Kummer-type* approaches to the embedding problem for prime power degree extensions. We briefly review the works of

Lenstra [36], and Allombert [2, 3], then we give variants of these algorithms with significantly lower complexities. As stated above, we let k, K be degree r extensions of $\mathbb{F}_q$, where r is a prime power, and we let p be their characteristic. We give our fast versions of the algorithms for two separate cases: the case $p \nmid r$ is treated in Section 3.1, the case $r = p^d$, where d is a positive integer, is treated in Section 3.2. Finally, in Section 3.3 we give a variant of the case $p \nmid r$ better suited for the case where r is a high-degree prime power.

In [36], Lenstra proves that given two finite fields of the same size, there exists a deterministic polynomial time algorithm that finds an isomorphism between them. The focus of the paper is on theoretical computational complexity; in particular, it avoids using randomized subroutines, such as polynomial factorization. In [2, 3], Allombert gives a similar approach with more focus on practical efficiency. In contrast to Lenstra's, his algorithm relies on polynomial factorization, thus it is polynomial time Las Vegas. Even though neither of the two algorithms is given a detailed complexity analysis, both rely on solving linear systems, thus a rough analysis yields an estimate of $O(r^\omega)$ operations in $\mathbb{F}_q$ in both cases.

The idea of Lenstra's algorithm is as follows. Assume that r is prime, and let $\mathbb{F}_q[\zeta], k[\zeta]$ denote the ring extensions $\mathbb{F}_q[Z]/\Phi_r(Z), k[Z]/\Phi_r(Z)$, where Φ_r is the rth cyclotomic polynomial. From a normal basis of k, computed using linear algebra, Lenstra constructs an element $\theta_1 \in k[\zeta]$ such that θ_1 and $\tau_1 = \theta_1^r$ are generators of the *Teichmüller subgroups* of $k[\zeta]$ and $\mathbb{F}_q[\zeta]$, respectively. He then proves that there is an $\mathbb{F}_q[\zeta]$-isomorphism of rings

$$k[\zeta] \xrightarrow{\sim} \mathbb{F}_q[\zeta][Y]/(Y^r - \tau_1)$$

sending θ_1 to the class of Y. Doing the same for $K[\zeta]$, elements $\theta_2 \in K[\zeta]$ and $\tau_2 \in \mathbb{F}_q[\zeta]$ and an $\mathbb{F}_q[\zeta]$-isomorphism

$$\mathbb{F}_q[\zeta][Y]/(Y^r - \tau_2) \xrightarrow{\sim} K[\zeta],$$

that sends the class of Y to θ_2, is obtained. Since τ_1 and τ_2 both generate the Teichmüller subgroup of $\mathbb{F}_q[\zeta]$, there exists an integer $j > 0$ such that $\tau_1 = \tau_2^j$, then the map

$$\begin{array}{rccc} \psi: & k[\zeta] & \to & K[\zeta] \\ & \theta_1 & \mapsto & \theta_2^j \end{array}$$

is an isomorphism of rings. Finally, denoting by Δ the automorphism group of $k[\zeta]$ over k, an embedding $k \hookrightarrow K$ is obtained by restricting the above isomorphism ψ to the fixed field $k[\zeta]^\Delta$. To summarize, the algorithm is made of three steps:

- Construct elements $\theta_1 \in k[\zeta]$ and $\theta_2 \in K[\zeta]$;
- Letting $\tau_i = \theta_i^r$, find the integer j such that $\tau_1 = \tau_2^j$ by a discrete logarithm computation in $\mathbb{F}_q[\zeta]$;
- Compute $\alpha \in k$ and $\beta \in K$ as some functions of θ_1, θ_2^j invariant under Δ.

The algorithm is readily generalized to prime powers r by iterating this procedure.

Allombert's algorithms differ from Lenstra's in two key steps, both resorting to polynomial factorization. First, he computes an irreducible factor h of the cyclotomic polynomial Φ_r of degree s, and so constructs a field extension $\mathbb{F}_q(\zeta)$ as $\mathbb{F}_q[Z]/h(Z)$. Then he defines $k[\zeta] = k[Z]/h(Z)$ and $K[\zeta] = K[Z]/h(Z)$ (note that these are not fields if r is not prime), and constructs $\theta_1 \in k[\zeta]$ and $\theta_2 \in K[\zeta]$ in a way equivalent to Lenstra's using linear algebra. At this point, rather than computing a discrete logarithm, Allombert points out that there exists a $c \in \mathbb{F}_q(\zeta)$

such that $\theta_1 \mapsto c\theta_2$ defines an isomorphism, and that such value can be computed as the rth root of θ_1^r/θ_2^r. Finally, by making the automorphism group of $k[\zeta]$ over k act on θ_1 and θ_2, he obtains an embedding $k \hookrightarrow K$.

3.1. **Allombert's algorithm.**

In this section, we analyze the complexity of Allombert's original algorithm [2], that of its revised version [3], and we present new variants with the best known asymptotic complexities. The main difference with respect to the versions presented in [2,3] is in the way we compute θ_1, θ_2, which are solutions to Hilbert's theorem 90 as will become clear below. Whereas Allombert resorts to linear algebra, we rely instead on evaluation formulas that have a high probability of yielding a solution. Recently, Narayanan [42, Sec. 3] independently described a variant which is similar to our Proposition 8 in the special case $s = 1$.

3.1.1. *General strategy.* Let $k = \mathbb{F}_q[X]/f(X)$ where f has degree r, a prime power, and let x be the image of X in k. Let $h(Z)$ be an irreducible factor of the rth cyclotomic polynomial over $\mathbb{F}_q$. Then h has degree s where s is the order of q in the multiplicative group $(\mathbb{Z}/r\mathbb{Z})^\times$. We form the field extension $\mathbb{F}_q(\zeta) \cong \mathbb{F}_q[Z]/h(Z)$ and the ring extension $k[\zeta] = k[Z]/h(Z) \cong k \otimes \mathbb{F}_q(\zeta)$ where ζ is the image of Z in the quotients. The action of the Galois group $\mathrm{Gal}(k/\mathbb{F}_q)$ can be extended to $k[\zeta]$ by

$$\begin{array}{rccc} \sigma: & k[\zeta] & \to & k[\zeta], \\ & x \otimes \zeta & \mapsto & x^q \otimes \zeta. \end{array}$$

Allombert shows (see [2, Prop. 3.2]) that σ is an automorphism of $\mathbb{F}_q(\zeta)$-algebras, and that its fixed set is isomorphic to $\mathbb{F}_q(\zeta)$. The same can be done for the ring $K[\zeta]$. Let us restate the algorithm for clarity.

Algorithm 3. Allombert's algorithm

```
Input: Field extensions $k, K$ of $\mathbb{F}_q$ of degree $r$.
Output: The description of a field embedding $k \to K$.
 1. Factor the $r$th cyclotomic polynomial and make the extensions $\mathbb{F}_q(\zeta), k[\zeta]$, $K[\zeta]$;
 2. Find $\theta_1 \in k[\zeta]$ such that $\sigma(\theta_1) = \zeta\theta_1$;
 3. Find $\theta_2 \in K[\zeta]$ such that $\sigma(\theta_2) = \zeta\theta_2$;
 4. Compute an $r$th root $c$ of $\theta_1^r/\theta_2^r$ in $\mathbb{F}_q(\zeta)$;
 5. Let $\alpha, \beta$ be the constant terms of $\theta_1, c\theta_2$, respectively;
 6. return The field embedding defined by $\alpha \mapsto \beta$.
```

The cyclotomic polynomial Φ_r is factored over $\mathbb{F}_q$ using [52, Theorem 9], and rth root extraction in $\mathbb{F}_q(\zeta)$ is done using Proposition 4, so we are left with the problem of finding θ_1 (and θ_2), that is, instances of Hilbert's theorem 90.

We now show how to do it in the extension $k[\zeta]/\mathbb{F}_q(\zeta)$, the case of $K[\zeta]$ being analogous. We review approaches due to Allombert, that rely on linear algebra, and propose new algorithms that rely on evaluation formulas and ultimately polynomial arithmetic. Note that all these variants can be directly applied to any extension degree r as long as $p \nmid r$, and do not require r to be a prime power. Nevertheless, in practice, it is more efficient to perform computations for each primary factor independently and glue the results together in the end.

If A is a polynomial with coefficients in $\mathbb{F}_q(\zeta)$, we will denote by $\hat{A}$ the morphism $A(\sigma)$ of the algebra $k[\zeta]$; note that the usual property of *q-polynomials* holds: $\widehat{AB} = \hat{A} \circ \hat{B}$.

3.1.2. *Algorithms relying on linear algebra.* As some algorithmic details were omitted in Allombert's publications, and no precise complexity analysis was performed, we extracted the details from PARI/GP source code [54] and perform the complexity analysis here. We also propose another variant, using an algorithm by Paterson and Stockmeyer.

Allombert's original algorithm. A direct solution to Hilbert's theorem 90 is to find a non-zero $\theta \in k[\zeta]$ such that $\widehat{(S-\zeta)}(\theta) = 0$.

The original version of Allombert's algorithm [2] does precisely this, by computing the matrix of the Frobenius automorphism σ of $k/\mathbb{F}_q$ using $O(\mathsf{M}(r)\log(q) + r\mathsf{M}(r))$ operations in $\mathbb{F}_q$ and then an eigenvalue of σ for ζ over $\mathbb{F}_q(\zeta)$ using linear algebra, at a cost of $O((rs)^\omega)$ operations in $\mathbb{F}_q$. This gives a total cost of $O(s\mathsf{M}(r)\log(q) + (rs)^\omega)$ operations in $\mathbb{F}_q$.

Allombert's revised algorithm. Allombert's revision of his own algorithm [3] uses the factorization

$$h(S) = (S-\zeta)b(S). \tag{2}$$

If we set $h(S) = S^s + \sum_{i=0}^{s-1} h_i S^i$, we can explicitly write b as

$$b(S) = \sum_{i=0}^{s-1} b_i(S)\zeta^i, \quad \text{where} \quad \begin{cases} b_{s-1}(S) = 1, \\ b_{i-1}(S) = b_i(S)S + h_i. \end{cases} \tag{3}$$

Indeed, Horner's rule shows that $b_{-1}(S) = h(S)$, and by direct calculation we find that $(S-\zeta)\cdot b(S) = b_{-1}(S)$.

We get a solution to Hilbert's theorem 90 by evaluating $b(S) = h(S)/(S-\zeta)$ on an element in the kernel of $\hat{h}$ over k, linear algebra now taking place over $\mathbb{F}_q$ rather than $\mathbb{F}_q(\zeta)$. The details on the computation of $\hat{h}$ were extracted from PARI/GP source code and yield the following complexity.

Proposition 6. *Using Allombert's revised algorithm, a solution θ to Hilbert's theorem 90 can be computed in $O(\mathsf{M}(r)\log(q) + sr\mathsf{M}(r) + r^\omega)$ operations in $\mathbb{F}_q$.*

Proof. As in Allombert's original algorithm, one first computes the matrix of σ over k at a cost of $O(\mathsf{M}(r)\log(q) + r\mathsf{M}(r))$ operations in $\mathbb{F}_q$.

To get the matrix of $\hat{h}$ over k, one first computes the powers x^{q^i} for $0 \le i \le s$ using the matrix of σ, at a cost of $O(sr^2)$ operations in $\mathbb{F}_q$. From them, one can iteratively compute the powers x^{jq^i} for $2 \le j \le r$ for a total cost of $O(sr\mathsf{M}(r))$ operations in $\mathbb{F}_q$, and iteratively compute the matrix of $\hat{h}$ for an additional total cost of $O(sr^2)$ operations in $\mathbb{F}_q$, accounting for the scalar multiplications by the coefficients of h. The total cost is therefore dominated by $O(sr\mathsf{M}(r))$ operations in $\mathbb{F}_q$.

Given the matrix of $\hat{h}$ over k, computing an element in its kernel costs $O(r^\omega)$ operations in $\mathbb{F}_q$. The final evaluation of $\hat{b}$ is done using (3) and the matrix of σ for Frobenius computations, for a cost of $O(sr^2)$ operations in $\mathbb{F}_q$. □

Using the Paterson–Stockmeyer algorithm. Given the matrix M_σ of σ, there is a natural way of evaluating $\hat{h}$ at a reduced cost: the Paterson–Stockmeyer algorithm [45] computes the matrix of $\hat{h}$ and $h(M_\sigma)$, using $O(\sqrt{s}r^\omega)$ operations in $\mathbb{F}_q$. The evaluations of σ that take a total of $O(sr^2)$ operations in $\mathbb{F}_q$ can be done directly using modular exponentiations, for a total of $O(s\mathsf{M}(r)\log(q))$.

Proposition 7. *Using the Paterson–Stockmeyer algorithm and modular exponentiations, a solution θ to Hilbert's theorem 90 can be computed in $O(s\mathsf{M}(r)\log(q) + \sqrt{s}r^{\omega})$ operations in $\mathbb{F}_q$.*

Although this complexity is not as good as the ones we will obtain next, this variant performs reasonably well in practice, as discussed in Section 7.

3.1.3. *Algorithms relying on polynomial arithmetic.* It is immediate to see that the minimal polynomial of σ over $k[\zeta]$ is $S^r - 1$; by direct calculation, we verify that it factors as

$$S^r - 1 = (S - \zeta) \cdot \Theta(S) = (S - \zeta) \sum_{i=0}^{r-1} \zeta^{-i-1} S^i. \tag{4}$$

Hence, we can set

$$\theta_a = \hat{\Theta}(a) = a \otimes \zeta^{-1} + \sigma(a) \otimes \zeta^{-2} + \cdots + \sigma^{r-1}(a) \otimes \zeta^{-r} \tag{5}$$

for some $a \in k$ chosen at random. Because of (4), θ_a is a solution as long as it is non-zero. This is reminiscent of Lenstra's algorithm [36, Th. 5.2].

To ensure the existence of a such that $\theta_a \neq 0$, we only need to prove that k is not entirely contained in $\ker \hat{\Theta}$. But the maps σ^i restricted to k are all distinct, thus Artin's theorem on character independence (see [33, Ch VI, Theorem 4.1]) shows that they are linearly independent, and therefore $\hat{\Theta}$ is not identically zero on k. In practice, we take $a \in k$ at random until $\theta_a \neq 0$. Since the map $\hat{\Theta}$ is $\mathbb{F}_q$-linear and non-zero, it has rank at least 1, thus a random θ_a is zero with probability less than $1/q$. Therefore, we only need $O(1)$ trials to find θ_1 (and θ_2).

Using the polynomial $b(S)$ introduced in (2), and defining $g(S) = (S^r - 1)/h(S)$, we can rewrite (4) as

$$\Theta(S) = b(S) \cdot g(S). \tag{6}$$

Then, the morphism $\hat{\Theta}$ can be evaluated as $\hat{b} \circ \hat{g}$, the advantage being that g has coefficients in $\mathbb{F}_q$, rather than in $\mathbb{F}_q(\zeta)$: we set $\tau_a = \hat{g}(a)$ for some $a \in k$ chosen at random and compute $\theta_a = \hat{b}(\tau_a)$ using (3), yielding a solution to Hilbert's theorem 90 as soon as $\tau_a \neq 0$. As before, $O(1)$ trials are enough to get $\theta_a \neq 0$.

We now give three variations on the above algorithm to compute a candidate solution θ_a more efficiently. Which algorithm has the best asymptotic complexity depends on the value of s with respect to r; we arrange them by increasing s.

First solution: divide-and-conquer recursion. We use a recursive algorithm similar to the computation of trace-like functions in Proposition 2, to directly evaluate θ_a using (5). For $j \geq 1$, let $\xi_j = \sigma^j(x)$ and $\theta_{a,j} = a \otimes \zeta^{-1} + \sigma(a) \otimes \zeta^{-2} + \cdots + \sigma^{j-1}(a) \otimes \zeta^{-j}$, so that we want to compute $\theta_{a,r} = \theta_a$. For $j = 1$, we have $(\xi_1, \theta_{a,1}) = (x^q, a\zeta^{-1})$; for $i, j \geq 1$, we have the following recursive relations:

$$(\xi_{i+j}, \theta_{a,i+j}) = (\sigma^i(\xi_j), \theta_{a,i} + \sigma^i(\theta_{a,j})\zeta^{-i}). \tag{7}$$

Proposition 8. *Given $a \in k$, the value θ_a in (5) can be computed using*

$$O(s^{(\omega-1)/2} r^{(\omega+1)/2} \log(r) + \mathsf{M}(r)\log(q))$$

operations in $\mathbb{F}_q$.

Proof. The value ξ_1 is computed by binary powering using $O(\mathsf{M}(r)\log(q))$ operations, while the value $\theta_{a,1}$ is deduced from the polynomial h using $O(rs)$ operations.

To compute the recursive formulas in (7) we use the same technique as in Proposition 2: given $b \in k[\zeta]$, the value $\sigma^j(b)$ is computed as the modular composition of the polynomial $b(x, z)$ with the polynomial $\xi_j(x)$ in the first argument. Each modular composition in $k[\zeta]$ is done using s modular compositions in k, at a cost of $O(s^{(\omega-1)/2}r^{(\omega+1)/2})$ operations (see Note 1). Multiplications by ζ^{-j} are done by seeing the elements of $k[\zeta]$ as polynomials in x over $\mathbb{F}_q(\zeta)$, thus performing r multiplications modulo h, at a cost of $O(r\mathsf{M}(s))$ operations. Given that the total depth of the recursion is $O(\log(r))$, we obtain the stated bound. □

Second solution: automorphism evaluation. We use (6) and (3) to compute θ_a as $\theta_a = \hat{b} \circ \hat{g}(a)$.

Proposition 9. *Given $a \in k$, the value θ_a in* (5) *can be computed using*

$$O(r^{(\omega^2-4\omega-1)/(\omega-5)} + (s + r^{2/(5-\omega)})\mathsf{M}(r)\log(q))$$

operations in $\mathbb{F}_q$.

Proof. We proceed in two steps. We first compute $\hat{g}(a)$ using the *automorphism evaluation* algorithm of Kaltofen and Shoup [24, Algorithm AE], at a cost of $O(r^{(\omega+1)/2+(3-\omega)|\beta-1/2|} + r^{(\omega+1)/2+(1-\beta)(\omega-1)/2} + r^{\beta}\mathsf{M}(r)\log(q))$, for any $0 \le \beta \le 1$. Choosing $\beta = 2/(5-\omega)$ minimizes the overall runtime, giving the exponents reported above.

We then use (3) to compute $\theta_a = \sum_{i=0}^{s-1} a_i \otimes \zeta^i$, where $a_{s-1} = \hat{g}(a)$, and $a_{i-1} = \sigma(a_i) + h_i\hat{g}(a)$. The cost of this computation is dominated by the evaluations of σ, which take $O(\mathsf{M}(r)\log(q))$ operations each, thus contributing $O(s\mathsf{M}(r)\log(q))$ total operations. □

Third solution: multipoint evaluation. Finally, we can compute all the values $\sigma(a), \dots, \sigma^{r-1}(a)$ directly, write θ_a as a polynomial in x and ζ of degree $r-1$ in both variables, and reduce modulo h for each power x^i.

Proposition 10. *Given $a \in k$, the value θ_a in* (5) *can be computed using*

$$O(\mathsf{M}(r^2)\log(r) + \mathsf{M}(r)\log(q))$$

operations in $\mathbb{F}_q$.

Proof. The values $\sigma(a), \dots, \sigma^{r-1}(a)$ can be computed by binary powering using $O(r\mathsf{M}(r)\log(q))$. We can do slightly better using the iterated Frobenius technique of von zur Gathen and Shoup [56, Algorithm 3.1] (see also [55, Ch. 14.7]), which costs of $O(\mathsf{M}(r^2)\log(r) + \mathsf{M}(r)\log(q))$ operations. The final reduction modulo h costs $O(r\mathsf{M}(r)\log(r))$ operations, which is negligible in front of the previous step. □

The following proposition summarizes our analysis. To clarify the order of magnitude of the exponents, let us assume $q = O(1)$ and neglect polylogarithmic factors; then, if $\omega = 2.38$ (best bound to date), the runtimes are $O(s^{0.69}r^{1.69})$ for $s \in O(r^{0.23})$, $O(r^{1.85} + s^{1.38}r)$ for $s \in \Omega(r^{0.23})$ and $s \in O(r^{0.72})$, and $\tilde{O}(r^2)$ otherwise. For $\omega = 3$, all costs are at best quadratic.

Proposition 11. *Given k, K of degree r over $\mathbb{F}_q$, assuming that s is the order of q in $(\mathbb{Z}/r\mathbb{Z})^\times$, Algorithm 3 computes its output using:*

- $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r) + \mathsf{M}(r)\log(q))$ *expected operations in* $\mathbb{F}_q$ *if* $s \in O(r^{(\omega-3)/(\omega-5)})$, *or*
- $O(r^{(\omega^2-4\omega-1)/(\omega-5)} + (s + r^{2/(5-\omega)})\mathsf{M}(r)\log(q) + s^{\omega-1}r\log(r)\log(s))$ *expected operations in* $\mathbb{F}_q$ *if if* $s \in \Omega(r^{(\omega-3)/(\omega-5)})$ *and* $s \in O(r^{1/(\omega-1)})$, *or*
- $O(\mathsf{M}(r^2)\log^2(r) + \mathsf{M}(r)\log(r)\log(q))$ *expected operations in* $\mathbb{F}_q$ *otherwise.*

Proof. The cost of factoring the rth cyclotomic polynomial is an expected $O(\mathsf{M}(r)\log(rq))$ operations in $\mathbb{F}_q$, using [52, Theorem 9]. This is negligible compared with other steps. The solutions θ_1, θ_2 to Hilbert's theorem 90 are computed as described above, according to the size of s. The powers θ_1^r, θ_2^r are computed using Kronecker substitution in $O(\mathsf{M}(sr)\log(r))$ operations, which is also negligible. Finally, the cost of computing an rth root in $\mathbb{F}_q(\zeta)$ is given by Proposition 4 and can not be neglected.

Combining the costs coming from the solution to Hilbert's theorem 90 and the rth root extraction, we obtain the following complexities according to s.

- If we use the algorithm described in our first solution, combining Proposition 8 with the first case of Proposition 4, we obtain an estimate of $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r) + \mathsf{M}(r)\log(q))$ operations.
- If we use the algorithm described in our second solution, combining Proposition 9 with the first case of Proposition 4, we obtain an estimate of $O(r^{(\omega^2-4\omega-1)/(\omega-5)} + (s + r^{2/(5-\omega)})\mathsf{M}(r)\log(q) + s^{\omega-1}r\log(r)\log(s) + \mathsf{M}(rs)\log(r)\log(s))$ operations.
- Otherwise, we use the algorithm described in our third solution. Combining Proposition 10 with the second case of Proposition 4, and replacing s with r everywhere, we obtain an estimate of $O(\mathsf{M}(r^2)\log^2(r) + \mathsf{M}(r)\log(r)\log(q))$ expected operations.

For $s \in O(r^{(\omega-3)/(\omega-5)})$, the first solution has the better runtime. Assuming $s \in \Omega(r^{(\omega-3)/(\omega-5)})$, the runtime in the second case can be written as $O(r^{(\omega^2-4\omega-1)/(\omega-5)} + (s + r^{2/(5-\omega)})\mathsf{M}(r)\log(q) + s^{\omega-1}r\log(r)\log(s))$. If in addition s is in $O(r^{1/(\omega-1)})$, this runtime is subquadratic, that is, better than that in our third solution. □

3.2. **The Artin–Schreier case.**

This section is devoted to the case $r = p^d$ for some positive integer d. The technique we present here originates in Adleman and Lenstra's work [1, Lemma 5], and appears again in Lenstra's [36] and Allombert's [2]. The chief difference with previous work once again consists in replacing linear algebra with a technique to solve the additive version of Hilbert's theorem 90 similar to the one in the previous section. Recently, Narayanan [42, Sec. 4] independently described a related variant with a similar complexity.

The idea is to build a tower inside the extension $k/\mathbb{F}_q$ using polynomials of the form $X^p - X - a$, where $a \in k$. To start, let $a_1 \in \mathbb{F}_q$ be such that $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_p}(a_1) \neq 0$. Let $\sigma \in \mathrm{Gal}(\mathbb{F}_q/\mathbb{F}_p)$ be a generator of the Galois group. Then by the additive version of Hilbert's theorem 90 there is no element $\alpha \in \mathbb{F}_q$ such that $\sigma(\alpha) - \alpha = a_1$. Equivalently, the polynomial $f_1 = X^p - X - a_1$ has no root in $\mathbb{F}_q$. By the Artin–Schreier theorem in [33, Ch VI], f_1 is irreducible over $\mathbb{F}_q$. For a root $\alpha_1 \in k$ of f_1, the extension $\mathbb{F}_q(\alpha_1)/\mathbb{F}_q$ is of degree p. Now let $a_2 = a_1\alpha_1^{p-1}$. Then, by [1, Lemma 5], the polynomial $f_2 = X^p - X - a_2$ is irreducible over $\mathbb{F}_q(\alpha_1)$. So, for a root $\alpha_2 \in k$ of f_2 the extension $\mathbb{F}_q(\alpha_2, \alpha_1)/\mathbb{F}_q(\alpha_1)$ is of degree p. Continuing the above process we build a tower

$$\mathbb{F}_q \subset \mathbb{F}_q(\alpha_1) \subset \cdots \subset \mathbb{F}_q(\alpha_1, \ldots, \alpha_d) = k. \tag{8}$$

The idea of building such tower using the Artin–Schreier polynomials f_i can also be found in [2,36,51]. By construction, $\alpha_i \notin \mathbb{F}_q(\alpha_1, \ldots, \alpha_{i-1})$ for all $1 \leq i \leq d$. This means that the minimal polynomial of α_d over $\mathbb{F}_q$ is of degree $r = p^d$. Therefore, $k = \mathbb{F}_q(\alpha_d)$, and the element α_d is uniquely defined up to $\mathbb{F}_q$-isomorphism.

The above construction boils down to computing a root of the polynomial $f = X^p - X - a \in k[X]$. We now show how to efficiently compute such a root. By construction, a is always in an intermediate subfield $\mathbb{F}_q \subseteq k' \subset k$. This means

$$\mathrm{Tr}_{k/\mathbb{F}_p}(a) = \mathrm{Tr}_{k'/\mathbb{F}_p}(\mathrm{Tr}_{k/k'}(a)) = \mathrm{Tr}_{k'/\mathbb{F}_p}(p^i a) = 0$$

for some $i > 0$. By Hilbert's theorem 90 there exists $\alpha \in k$ such that $\alpha - \sigma(\alpha) = -a$ for a generator $\sigma \in \mathrm{Gal}(k/\mathbb{F}_p)$. In other words, $\alpha^p - \alpha - a = 0$. Therefore, α is a root of f. On the other hand, for a random element $\theta \in k$ with non-zero trace, α can be explicitly set as

$$\alpha = \frac{1}{\mathrm{Tr}(\theta)}[a\sigma(\theta) + (a + \sigma(a))\sigma^2(\theta) + \cdots + (a + \sigma(a) + \cdots + \sigma^{rt-2}(a))\sigma^{rt-1}(\theta)], \tag{9}$$

where $t = [\mathbb{F}_q : \mathbb{F}_p]$. To compute α using (9) efficiently, for $j \geq 1$, we define $\xi_j = \sigma^j(x)$, and, for u in k,

$$\beta_{u,j} = u + \sigma(u) + \cdots + \sigma^{j-1}(u), \quad \alpha_{u,j} = \beta_{a,1}\sigma(u) + \cdots + \beta_{a,j}\sigma^j(u).$$

Our goal is to ultimately compute $\beta_{\theta,rt} = \mathrm{Tr}(\theta)$ and $\alpha = \beta_{\theta,rt}^{-1}\alpha_{\theta,rt}$. For this, we show how to compute quadruples of the form $(\xi_j, \beta_{a,j}, \beta_{\theta,j}, \alpha_{\theta,j})$, for $j \geq 1$.

For $j = 1$, we have $(\xi_1, \beta_{a,1}, \beta_{\theta,1}, \alpha_{\theta,1}) = (x^q, a, \theta, a\theta^q)$; for $i, j \geq 1$, a calculation gives

$$\begin{aligned}(10)\quad &(\xi_{i+j}, \beta_{a,i+j}, \beta_{\theta,i+j}, \alpha_{\theta,i+j})\\ &= \bigl(\sigma^j(\xi_i),\ \beta_{a,j} + \sigma^j(\beta_{a,i}),\ \beta_{\theta,j} + \sigma^j(\beta_{\theta,i}),\ \alpha_{\theta,j} + \sigma^j(\alpha_{\theta,i}) + \beta_{a,j}\sigma^{j+1}(\beta_{\theta,i})\bigr).\end{aligned}$$

In particular, the values $\beta_{\theta,rt}$, and $\alpha_{\theta,rt}$ can be computed recursively, in $O(\log(rt))$ steps. The initial values $(\xi_1, \beta_{a,1}, \beta_{\theta,1}, \alpha_{\theta,1})$ are computed using $O(\mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$. For higher indices, as before, the action of σ^j is the same as composing with ξ_j, so each step of the recursion is dominated by $O(1)$ modular compositions over $\mathbb{F}_q$, at the cost of $O(r^{(\omega+1)/2})$ operations in $\mathbb{F}_q$. Therefore, the cost of computing a root of f is $O(r^{(\omega+1)/2}\log(rt) + \mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$.

Now, to compute α_d in (8) we need to take d roots where $d \in O(\log(r)/\log(p))$ which leads to the following result. (Note that ξ_1 is computed only once and reused thereafter.)

Proposition 12. *Let $r = p^d$ for a positive integer d, and let $t = [\mathbb{F}_q : \mathbb{F}_p]$. An isomorphism of two extensions $k/\mathbb{F}_q$, $K/\mathbb{F}_q$ of degree r can be constructed using $O(r^{(\omega+1)/2}\log(rt)\log(r) + \mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$.*

3.3. **High-degree prime powers.** We end this section with an algorithm that is particularly efficient when the extension degree r is a high-degree prime power. Allombert's algorithm works well in this case, however, its complexity depends linearly on the order s of q modulo r. If $r = v^d$ for some prime $v \neq p$, it is natural to seek an algorithm which depends on the order of q modulo v instead. The idea we present is a variation on Lenstra's algorithm, using successive vth root extractions. We are not aware of this algorithm appearing anywhere in the literature. We also note that Narayanan [42, Sec. 5] recently published a radically different generalization of Allombert's algorithm with a very similar complexity in r (his algorithm has much worse complexity in q, though).

An overview of our construction is as follows. Let $r = v^d$ where $v \neq p$ is a prime and d is a positive integer. Suppose the extension $k/\mathbb{F}_q$ is of degree r. Let s be the order of q in $\mathbb{Z}/v\mathbb{Z}$, and write $q^s - 1 = uv^t$ where $\gcd(v, u) = 1$. Since $q^r = q^{v^d} = q \bmod v$, we see that q^r has the same order s in $\mathbb{Z}/v\mathbb{Z}$. So the cyclotomic field extensions $\mathbb{F}_q(\zeta)/\mathbb{F}_q, k(\zeta)/k, K(\zeta)/K$ are all of degree s. We move to these extensions by obtaining an irreducible factor of the vth cyclotomic polynomial over $\mathbb{F}_q$. Next, we obtain a random non-vth power η in $\mathbb{F}_q(\zeta)^*$.

It follows from $v^t \mid q^s - 1$ that $v^{t+d} \mid q^{rs} - 1$. So we can compute an rth root θ_1 of η in $k(\zeta)$ using d successive vth root extractions in $k(\zeta)$. Since η is a non-vth power, the polynomial $Y^r - \eta$ is irreducible over $\mathbb{F}_q(\zeta)$ and there is an $\mathbb{F}_q(\zeta)$-isomorphism

$$\mathbb{F}_q(\zeta)(\theta_1) \xrightarrow{\sim} \mathbb{F}_q(\zeta)[Y]/(Y^r - \eta)$$

that sends θ_1 to Y. From this we see that $\mathbb{F}_q(\theta)$ and $k(\zeta)$ have the same degree over $\mathbb{F}_q$, and hence $\mathbb{F}_q(\theta_1) = k(\zeta)$. Similarly, we find $\theta_2 \in K(\zeta)$. So we have an $\mathbb{F}_q(\zeta)$-embedding

$$\phi : k(\zeta) \xrightarrow{\sim} \mathbb{F}_q(\zeta)[Y]/(Y^r - \eta) \xrightarrow{\sim} K(\zeta)$$

that sends θ_1 to θ_2. Let $\alpha = \mathrm{Tr}_{k(\zeta)/k}(\theta_1)$ and $\beta = \mathrm{Tr}_{K(\zeta)/K}(\theta_2)$. Then $k = \mathbb{F}_q(\alpha)$ and $K = F_q(\beta)$; see [51, Algorithm 13] and the proof of [50, Theorem 2.1]. Since ϕ commutes with traces, the map $\alpha \mapsto \beta$ defines an isomorphism.

The main difficulty in applying such an algorithm resides in computing efficiently vth roots in $k(\zeta)$, for which we use Proposition 5; this yields the main result of this section.

Theorem 13. *Let $r = v^d$, where $v \neq p$ is a prime and d is a positive integer. Also, let s be the order of q in $\mathbb{Z}/v\mathbb{Z}$. Given extensions $k/\mathbb{F}_q$, $K/\mathbb{F}_q$ of degree r, an embedding $k \hookrightarrow K$ can be constructed at the cost of an expected $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r)^2 + s\mathsf{M}(r)\log(r)\log(q)$ operations in $\mathbb{F}_q$.*

Proof. We can construct the embedding of Theorem 13 as follows. We first build the extensions $k(\zeta)/\mathbb{F}_q(\zeta)$ and $K(\zeta)/\mathbb{F}_q(\zeta)$. Let η be a non vth power in $\mathbb{F}_q(\zeta)$. Then η is an $r$power in $k(\zeta)$ and $K(\zeta)$. To obtain rth roots $\theta_1 \in k$, $\theta_2 \in K$ of η we take d successive vth roots.

Algorithm 4. Kummer-type algorithm for extension towers

```
Input: Extensions $k/\mathbb{F}_q$, $K/\mathbb{F}_q$ of degree prime-power $r = v^d$, with $v \neq p$.
Output: The description of a field embedding $k \hookrightarrow K$.
1. Factor the $v$th cyclotomic polynomial over $\mathbb{F}_q$ to build the extensions $k(\zeta)/\mathbb{F}_q(\zeta)$ and $K(\zeta)/\mathbb{F}_q(\zeta)$;
2. Find a random non-$v$th power $\eta \in \mathbb{F}_q(\zeta)$;
3. Compute $r$th roots $\theta_1, \theta_2$ of $\eta$ in $k(\zeta), K(\zeta)$;
4. Compute $\alpha = \mathrm{Tr}_{k(\zeta)/k}(\theta_1)$ and $\beta = \mathrm{Tr}_{K(\zeta)/K}(\theta_2)$;
5. return The field embedding defined by $\alpha \mapsto \beta$.
```

Step 1 is done using [52, Theorem 9], which takes $O(\mathsf{M}(v)\log(vq))$ operations in $\mathbb{F}_q$. We do Step 2 by taking random elements in $\mathbb{F}_q(\zeta)$ until a non-vth power is found. Testing whether η is a vth power amounts to computing $\eta^{(q^s-1)/v}$ in $\mathbb{F}_q(\zeta)$, which can be done in $O(s^{(\omega-1)/2}\log(s)+\mathsf{M}(s)\log(v)\log(s)+\mathsf{M}(s)\log(q))$ operations in $\mathbb{F}_q$, in view of our discussion in Section 2.

Step 3 is done using $d = O(\log(r)/\log(v))$ successive root extractions, each of which takes an expected $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r) + s\mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$. Writing θ_1 and θ_2 as polynomials in ζ with coefficients in k and K, respectively, we see that computing the traces of Step 4 boils down to computing the traces $\mathrm{Tr}_{\mathbb{F}_q(\zeta)/\mathbb{F}_q}(\zeta^i)$ for $0 \leq i < s$. The cost of this is dominated by those of the previous steps. Therefore, Algorithm 4 runs in an expected $O(s^{(\omega-1)/2}r^{(\omega+1)/2}\log(r)^2 + s\mathsf{M}(r)\log(r)\log(q)$ operations in $\mathbb{F}_q$. □

4. Rains' algorithm

We now move on to a different family of algorithms based on the theory of algebraic groups. The simplest of these is Pinch's cyclotomic algorithm [46]. The idea is very simple: given r, select an integer ℓ such that $[\mathbb{F}_q(\mu_\ell) : \mathbb{F}_q] = r$, where μ_ℓ is the group of ℓth roots of unity. Then, any embedding $k \to K$ takes $\mu_\ell \subset k^*$ to $\mu_\ell \subset K^*$, and the minimal polynomial of any primitive ℓth root of unity has degree exactly r.

Pinch's algorithm is very effective when $r = \varphi(\ell)$. Indeed, in this case the ℓth cyclotomic polynomial Φ_ℓ is irreducible over $\mathbb{F}_q$, and its roots form a unique orbit under the action of the absolute Galois group of $\mathbb{F}_q$. Thus we can take any primitive ℓth roots of unity $\alpha \in k$ and $\beta \in K$ to describe the embedding.

In the general case, however, the roots of Φ_ℓ are partitioned in $\varphi(\ell)/r$ orbits, thus for two randomly chosen ℓth roots of unity $\zeta_1 \in k$ and $\zeta_2 \in K$, we can only say that there exists an exponent e such that

$$\alpha = \zeta_1 \mapsto \zeta_2^e = \beta$$

defines a valid embedding. Pinch's algorithm tests all possible exponents e, until a suitable one is found. To test for the validity of a given e, it applies the embedding $\phi : \zeta_1 \mapsto \zeta_2$ to the class of X in k, and verifies that its image is a root of f in K.

The trial and error nature of Pinch's algorithm makes it impractical, except for rare favorable cases where a *small* ℓ such that $r = \varphi(\ell)$ can be found. One possible workaround, suggested by Pinch himself, is to replace the group of roots of unity with a group of torsion points of a well chosen elliptic curve. We analyze this idea in greater detail in Section 5.

This section is devoted to a different way of improving Pinch's algorithm, imagined by Rains [48], and implemented in the Magma computer algebra system [4]. Rains' technical contribution is twofold: first he replaces roots of unity with Gaussian periods to avoid trial and error, and second he moves to slightly larger extension fields to ensure the existence of a small ℓ as above.

4.1. **Uniquely defined orbits from Gaussian periods.** For the rest of the section, we are going to assume that q is prime. The case where q is a higher power of a prime is discussed in Note 18.

Suppose that we have an ℓ, coprime with q, such that $[\mathbb{F}_q(\mu_\ell) : \mathbb{F}_q] = r$, then the cyclotomic polynomial Φ_ℓ factors over $\mathbb{F}_q$ into $\varphi(\ell)/r$ distinct factors of degree r. Pinch's method, by choosing random roots of Φ_ℓ in k and K, randomly selects one of these factors as minimal polynomial. By combining the roots of Φ_ℓ into Gaussian periods, Rains' method uniquely selects a minimal polynomial of degree r.

Definition 14. Let q be a prime, and let ℓ be a squarefree integer such that $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle q\rangle \times S$ for some S. For any generator ζ_ℓ of μ_ℓ in $\mathbb{F}_q(\mu_\ell)$, define the Gaussian period $\eta_q(\zeta_\ell)$ as

$$\eta_q(\zeta_\ell) = \sum_{\sigma\in S} \zeta_\ell^\sigma. \tag{11}$$

It is evident from the definition that the Galois orbit of $\eta_q(\zeta_\ell)$ is independent of the initial choice of ζ_ℓ. Much less evident is the fact that this orbit has maximal size and forms a normal basis of $\mathbb{F}_q(\mu_\ell)$, as stated in the following lemma.

Lemma 15. *Let q be a prime, and let ℓ be a squarefree integer such that $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle q\rangle \times S$ for some S. The periods $\eta_q(\zeta_\ell^\tau)$ for τ running through $\langle q\rangle$ form a normal basis of $\mathbb{F}_q(\mu_\ell)$ over $\mathbb{F}_q$, independent of the choice of ζ_ℓ.*

Proof. Gaussian periods were introduced by Gauss [18] in 1796 and extensively studied by Kummer [26–32]. The existence of integral normal bases for general Galois extensions without ramification was proved by Noether [43]. For a generalized treatment specific to the case of finite fields; see [17, Main Theorem]: the main idea of the proof is to show that cyclotomic units are normal in characteristic zero, then that integrality conditions carry normality through reduction modulo q. □

In what follows we are going to write $\eta(\zeta_\ell)$ when q is clear from the context.

Example 16. Consider the extension $\mathbb{F}_8/\mathbb{F}_2$ of degree 3, which is generated by the 7th roots of unity. We have a decomposition $(\mathbb{Z}/7\mathbb{Z})^\times = \langle 2\rangle \times \langle -1\rangle$, and the cyclotomic polynomial factors as

$$\Phi_7(X) = (X^3 + X + 1)(X^3 + X^2 + 1). \tag{12}$$

For any root ζ_7, we define the period

$$\eta_2(\zeta_7) = \zeta_7 + \zeta_7^{-1}. \tag{13}$$

The three periods $\eta_2(\zeta_7)$, $\eta_2(\zeta_7)^2$ and $\eta_2(\zeta_7)^4$ are all roots of the polynomial $x^3 + x^2 + 1$ and form a normal basis of $\mathbb{F}_8/\mathbb{F}_2$.

4.2. **Rains' cyclotomic algorithm.** The bottom-line of Rains' algorithm follows immediately from the previous section: given k, K, and r,

(1) find a *small* ℓ satisfying the conditions of Lemma 15 with $[\mathbb{F}_q(\mu_\ell) : \mathbb{F}_q] = r$;
(2) take random ℓth roots of unity $\zeta_\ell \in k$ and $\zeta'_\ell \in K$;
(3) return the Gaussian periods $\alpha_r = \eta(\zeta_\ell)$ and $\beta_r = \eta(\zeta'_\ell)$.

The problem with this algorithm is the vaguely defined *smallness* requirement on ℓ. Indeed, the conditions of Lemma 15 imply that ℓ divides $\Phi_r(q)$, thus in the worst case ℓ can be as large as $O(q^{\varphi(r)})$, which yields an algorithm of exponential complexity in the field size.

To circumvent this problem, Rains allows the algorithm to work in small auxiliary extensions of k and K, and then descend the results to k and K via a field trace. In other words, Rains' algorithm looks for ℓ such that $[\mathbb{F}_q(\mu_\ell) : \mathbb{F}_q] = rs$ for some small s. We summarize this method in Algorithm 5; we only give the procedure for the field k, the procedure for the field K being identical.

Algorithm 5. Rains' cyclotomic algorithm

```
Input: A field extension $k/\mathbb{F}_q$ of degree $r$; a squarefree integer $\ell$ such that
        • $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle q \rangle \times S$ for some $S$,
        • $\#\langle q \rangle = rs$ for some integer $s$;
    a polynomial $h$ of degree $s$ irreducible over $k$.
Output: A normal generator of $k$ over $\mathbb{F}_q$, with a uniquely defined Galois orbit.
1. Construct the field extension $k' = k[Z]/h(Z)$;
2. repeat
3.     Compute $\zeta \leftarrow \theta^{(\#k'-1)/\ell}$ for a random $\theta \in k'$
4. until $\zeta$ is a primitive $\ell$th root of unity;
5. Compute $\eta(\zeta) \leftarrow \sum_{\sigma \in S} \zeta^\sigma$;
6. return $\alpha \leftarrow \mathrm{Tr}_{k'/k}\, \eta(\zeta) = \sum_{i=0}^{s-1} \eta(\zeta)^{q^{ri}}$.
```

Proposition 17. *Algorithm 5 is correct. On input q, r, ℓ, s it computes its output using $O(sr^{(\omega+1)/2} \log(sr) + \mathsf{M}(sr)(\log(q) + (\ell/r)\log(\ell)))$ operations in $\mathbb{F}_q$ on average.*

Proof. By construction k' is isomorphic to $\mathbb{F}_q(\mu_\ell)$. By Lemma 15 $\eta(\zeta)$ is a normal generator of k', and by [41, Prop. 5.2.3.1] α is a normal generator of k. This proves correctness.

According to Proposition 2, computing ζ in Step 3 costs

$$O\big(\big(s^{(\omega+1)/2}\mathsf{M}(r) + sr^{(\omega+1)/2} + \mathsf{M}(sr)\log(\ell)\big)\log(sr) + \mathsf{M}(sr)\log(q)\big),$$

and the loop is executed $O(1)$ times on average. By observing that $s^{(\omega-1)/2} \in O(\ell/r)$, this fits into the stated bound.

Steps 5 and 6 can be performed at once by observing that

$$\alpha = \sum_{i=0}^{s-1} \eta(\zeta^{q^{ri}}) = \sum_{i=0}^{s-1} \sum_{\sigma \in S} \zeta^{q^{ri}\sigma}.$$

By reducing $q^{ri}\sigma$ modulo ℓ, we can compute this sum at the cost of $\varphi(\ell)/r$ exponentiations of degree at most ℓ in k', for a total cost of $O((\mathsf{M}(sr)(\ell/r)\log(\ell))$, using the techniques of Section 2. The final result is obtained as an element of k. □

The attentive reader will have noticed the irreducible polynomial h of degree s given as input to Rains' algorithm. Computing this polynomial may be expensive.

For a start, we may ask s to be coprime with r, so that h can be taken with coefficients in $\mathbb{F}_q$. Then, for small values of s and q, one may use a table of irreducible polynomials. For larger values, the constructions [12–14] are reasonably efficient, and yield an irreducible polynomial in time less than quadratic in s. However, negligible from an asymptotic point of view, the construction of the polynomial h and of the field k' take a serious toll on the practical performances of Rains' algorithm.[4]

This concludes the presentation of Rains' algorithm. However, we are still left with a problem: how to find ℓ satisfying the conditions of the algorithm, and what bounds can be given on it. These questions will be analyzed in Section 6.

Note 18. Rains' algorithm is easily extended to a non-prime field $\mathbb{F}_q$, as long as $q = p^d$ with $\gcd(d, r) = 1$. In this case, indeed, any generator of $\mathbb{F}_{p^r}$ over $\mathbb{F}_p$ is also a generator of $\mathbb{F}_{q^r}$ over $\mathbb{F}_q$. The algorithm is unchanged, except for the additional requirement that $\gcd(\varphi(\ell), d) = 1$, which ensures that the Gaussian periods indeed generate $\mathbb{F}_{p^r}$.

However, when $\gcd(d, r) \neq 1$, it is impossible to have $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle q\rangle \times S$, so Rains' algorithm simply cannot be applied to this case. In the next section we are going to present a variant that does not suffer from this problem.

5. Elliptic Rains' algorithm

The Pinch–Rains' algorithm presented in the previous section relies on the use of the multiplicative group of finite fields. It is natural to try to extend it to other types of algebraic groups in order to cover a wider range of parameters. And indeed, Pinch [46] showed how to use torsion points of elliptic curves in place of roots of unity. Rains also considered this possibility, but did not investigate it thoroughly as no theoretical gain was to be expected. However, the situation in practice is quite different. In particular, the need for auxiliary extensions in the cyclotomic method is very costly, whereas the elliptic variant has naturally more chances to work in the base fields, and to be therefore very competitive.

In the next sections, we first introduce *elliptic periods*, a straightforward generalization of Gaussian periods for torsion points of elliptic curves, then analyze the cost of their computation. The main issue with this generalization is that, contrary to Gaussian periods, elliptic periods do not yield normal bases of finite fields. We still provide experimental data and heuristic arguments to support the benefit of using them. Whether they always yield an element generating the right field extension, a weak counterpart to Lemma 15, is left as an open problem.

5.1. Uniquely defined orbits from elliptic periods.

An elliptic curve E/L defined over a field L is given by an equation of the form

$$E\ :\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad \text{with } a_1, a_2, a_3, a_4, a_6 \in L.$$

For any field extension M/L the group of M-rational points of E is the set

$$E(M) = \{(x, y) \in M^2 \mid E(x, y) = 0\} \cup \{\mathcal{O}\}$$

endowed with the usual group law, where $\mathcal{O}$ is the point at infinity.

[4] A straightforward way to avoid these constructions consists in computing a factor h of the cyclotomic polynomial Φ_ℓ over the extension k following case 5 from Section 2.1. Then, using Newton's identities, the period can be recovered from the logarithmic derivative of the reciprocal of h. Nevertheless, the cost of factoring Φ_ℓ renders this approach unpractical.

For an integer ℓ, we denote by $E[\ell]$ the ℓ-torsion subgroup of $E(\bar{L})$, where $\bar{L}$ denotes the algebraic closure of L. In this section we are going to consider integers ℓ coprime with the characteristic of L, then $E[\ell]$ is a group of rank 2.

For an elliptic curve $E/\mathbb{F}_q$ defined over a finite field, we denote by π its *Frobenius endomorphism.* It is well known that π satisfies a quadratic equation $\pi^2-t\pi+q=0$, where t is called the *trace of* E, and that this equation determines the cardinality of E as $\#E(\mathbb{F}_q)=q+1-t$.

Like in the cyclotomic case, the Frobenius endomorphism partitions $E[\ell]$ into orbits. Our goal is to take traces of points in $E[\ell]$ so that a uniquely defined orbit arises. This task is made more complex by the fact that $E[\ell]$ has rank 2, hence we are going to restrict to a family of primes ℓ named *Elkies primes.*

Definition 19 (Elkies prime)**.** Let $E/\mathbb{F}_q$ be an elliptic curve, let ℓ be a prime number not dividing q. We say that ℓ is an Elkies prime for E if the characteristic polynomial of the Frobenius endomorphism π splits into two distinct factors over $\mathbb{Z}/\ell\mathbb{Z}$:

$$\pi^2-t\pi+q=(\pi-\lambda)(\pi-\mu) \mod \ell \qquad \text{with } \lambda\neq\mu. \tag{14}$$

Note that if ℓ is an Elkies prime for E, then $E[\ell]$ splits into two eigenspaces for π which are defined on extensions of $\mathbb{F}_q$ of degrees $\mathrm{ord}_\ell(\lambda)$ and $\mathrm{ord}_\ell(\mu)$. We are now ready to define the elliptic curve analogue of Gaussian periods.

Definition 20. Let $E/\mathbb{F}_q$ be an elliptic curve of j-invariant not 0 or 1728.[5] Let $\ell>3$ be an Elkies prime for E, λ an eigenvalue of π, and P a point of order ℓ in the eigenspace corresponding to λ (i.e., such that $\pi(P)=\lambda P$). Suppose that there is a subgroup S of $(\mathbb{Z}/\ell\mathbb{Z})^\times$ such that

$$(\mathbb{Z}/\ell\mathbb{Z})^\times=\langle\lambda\rangle\times S. \tag{15}$$

Then we define an elliptic period as

$$\eta_{\lambda,S}(P)=\begin{cases}\sum_{\sigma\in S/\{\pm1\}}x\left([\sigma]P\right) & \text{if } -1\in S,\\ \sum_{\sigma\in S}x\left([\sigma]P\right) & \text{otherwise,}\end{cases} \tag{16}$$

where $x(P)$ denotes the abscissa of P.

Lemma 21. *With the same notation as in Definition* 20*, let*

$$\#\langle\lambda\rangle=\begin{cases}r & \textit{if } -1\notin\langle\lambda\rangle,\\ 2r & \textit{otherwise.}\end{cases}$$

Then, for any point P in the eigenspace of λ, the period $\eta_{\lambda,S}(P)$ is in $\mathbb{F}_{q^r}$, and its minimal polynomial does not depend on the choice of P.

Proof. By construction, the Frobenius endomorphism π acts on $\langle P\rangle$ as multiplication by the scalar λ. It is well known that two points have the same abscissa if and only if they are opposite, hence the Galois orbit of $x(P)$ has size r, and we conclude that both $x(P)$ and $\eta_{\lambda,S}(P)$ are in $\mathbb{F}_{q^r}$.

Now let $P'=[a]P$ be another point in the eigenspace of λ. By construction, $a=\pm\lambda^i\sigma$, for some $0\le i<r$ and some $\sigma\in S$. Hence $\eta_{\lambda,S}(P')=\eta_{\lambda,S}([\lambda^i]P)$, implying that $\eta_{\lambda,S}(P)$ and $\eta_{\lambda,S}(P')$ are conjugates in $\mathbb{F}_{q^r}$. □

[5] The definition is easily extended to include $j=0,1728$: one must quotient S by $\mathrm{Aut}(E)\cap S$ and raise summands to an appropriate power. See [7] for more details.

We remark that the previous lemma only states that the elliptic periods $\eta_{\lambda,S}([\lambda^i]P)$ uniquely define an orbit inside $\mathbb{F}_{q^r}$, but gives no guarantee that they generate the whole $\mathbb{F}_{q^r}$. At this point, one would like to have an equivalent of Lemma 15 for elliptic periods, i.e., that the elliptic period $\eta_{\lambda,S}(P)$ is a normal generator of $\mathbb{F}_q(x(P))$. However, it is easy to find non-normal elliptic periods, as the following example shows.

Example 22. Let $E/\mathbb{F}_7$ be defined by $y^2 = x^3 + 5x + 4$, and consider the degree 3 extension of $\mathbb{F}_7$ defined by $k = \mathbb{F}_7[X]/(X^3 + 6X^2 + 4)$. Then

- $\ell = 31$ is an Elkies prime for E;
- the eigenvalues of the Frobenius modulo ℓ are $\lambda = 25$ of multiplicative order 3 and $\mu = 4$ of multiplicative order 5;
- $P = (5a^2 + 2a, 4)$ is a point of order 31 of E/k;
- $\eta = \eta_{\lambda,S}(P) = 5a^2+5a+4$ is not a normal element, indeed $\eta+4\eta^7+2\eta^{49} = 0$.

All known proofs of Lemma 15 rely on the fact that the ℓth cyclotomic polynomial is irreducible over $\mathbb{Q}$, and its roots form a normal basis of $\mathbb{Q}(\zeta_\ell)$. This fails in the elliptic case: there is indeed no guarantee that the eigenspace of λ can be lifted to a normal basis over some number field.

Note, however, that, even if the elliptic period is not normal, it is enough for our purpose that it generates $\mathbb{F}_q(x(P))$ as a field, like in the example above. Experimental evidence suggests that this might always be the case. Thus, we state this as a conjecture.

Conjecture 23. With the above notation, the elliptic period $\eta_{\lambda,S}(P)$ generates $\mathbb{F}_q(x(P))$ over $\mathbb{F}_q$.

If the conjecture is false, the only arguments we can give are of a heuristic nature. First and most simply, we can assume that the elliptic period behaves like a random element of $\mathbb{F}_q(x(P))$. In this case the chance of it not being a generator is approximately $1/q^r$. Based on this observation, numerous experiments were conducted for small values of q and r: for all primes ℓ up to 1000, and all primes q up to a bound depending on ℓ, we tested all curves defined over $\mathbb{F}_q$ such that ℓ is an Elkies prime and a corresponding eigenvalue r is an odd prime. Overall, more than 43 million curves were tested, and no counterexample was found. We also tested the conjecture through more involved methods using modular curves, more details are given in [7, Appendix C.2].[6]

Secondly, based on the polynomially cyclic algebras setting of [38], one can give a sufficient condition for the period to be a normal generator of $\mathbb{F}_q(x(P))$, that is, a weak counterpart to Lemma 15 (see [7, Appendix C.1]). Heuristically, this suggests that the chance of the period not being normal is approximately $1/q$.

We are now ready to present the generalization of Rains' algorithm, with the warning that the algorithm may fail, with low probability, if Conjecture 23 is false.

5.2. **Elliptic variant of Rains' algorithm.** Rain's cyclotomic algorithm needs auxiliary extensions to accommodate for sufficiently small subgroups μ_ℓ of the unit group. By replacing unit groups with torsion groups of elliptic curves, we gain more freedom on the choice of the size of the group, thus we are able to work with smaller fields.

[6] The source code for our tests is available at `https://github.com/defeo/ffisom/blob/master/misc/conjdata`.

The algorithm is very similar to Algorithm 5, and follows immediately from the previous section. For simplicity, we are going to state it only for r odd. Given k, K, and r,

(1) find a prime ℓ, an elliptic curve E, and an eigenvalue λ of the Frobenius endomorphism, satisfying the conditions of Definition 20, and such that $\operatorname{ord}_\ell(\lambda) = r$;
(2) take random points $P \in E(k)[\ell]$ and $P' \in E(K)[\ell]$ in the eigenspace of λ;
(3) return the elliptic periods $\alpha := \eta_{\lambda,S}(P)$ and $\beta := \eta_{\lambda,S}(P')$.

Here we are faced with a difficulty: given E and λ it is easy to pick a random point in $E[\ell]$, but it is potentially much more expensive to compute a point in the eigenspace of λ. We will circumvent the problem by forcing $E(\mathbb{F}_{q^r})[\ell]$ to be of rank 1, and to coincide exactly with the eigenspace of λ. If we write $\mu = q/\lambda$ for the other eigenvalue of π, this is easily ensured by further asking that $\operatorname{ord}_\ell(\mu) \nmid r$.

We defer the discussion on the search for the elliptic curve E to Section 6. Here we suppose that we are already given suitable parameters ℓ, E, and λ, and analyze the last two steps of the algorithm, summarized below. We only give the procedure for k, the procedure for the field K being identical.

Algorithm 6. Elliptic Rain's algorithm

Input: A field extension $k/\mathbb{F}_q$ of odd degree r, an elliptic curve $E/\mathbb{F}_q$, its trace t, a prime ℓ not dividing q, an integer λ such that:

- $X^2 - tX + q = (X-\lambda)(X-q/\lambda) \mod \ell$,
- $\operatorname{ord}_\ell(\lambda) = r$, $\operatorname{ord}_\ell(q/\lambda) \nmid r$,
- $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle\lambda\rangle \times S$ for some S.

Output: A generator of k over $\mathbb{F}_q$, with a uniquely defined Galois orbit, or FAIL.

```
repeat
    Compute $P \leftarrow [\#E(k)/\ell]Q$ for a random $Q \in E(k)$;
until $P \neq \mathcal{O}$;
Compute $\alpha \leftarrow \eta_{\lambda,S}(P)$;
return $\alpha$ if $k = \mathbb{F}_q(\alpha)$, FAIL otherwise.
```

Proposition 24. *Algorithm* 6 *is correct. Assuming the heuristics about elliptic periods are correct, it fails with probability* $\leq 1/q^r$. *On input* $r, q, E, t, \ell, \lambda$ *it computes its output using* $O(\mathsf{M}(r)(r\log(q) + (\ell/r)\log(\ell))$ *operations in* $\mathbb{F}_q$ *on average, or* $\tilde{O}(r^2\log(q))$ *assuming* $\ell \in o(r^2)$.

Proof. Correctness follows immediately from Lemma 21. Success probability comes from the assumption that $\eta_{\lambda,S}(P)$ behaves like a random element of $\mathbb{F}_q(x(P))$.

From the knowledge of the trace t, we immediately determine the zeta function of E, and hence the cardinality $\#E(k)$, at no algebraic cost.

To select the random point $Q \in E(k)$ we take a random element $x \in k$, then we verify that it is the abscissa of a point using a squareness test, at a cost of $O(r\mathsf{M}(r)\log(q))$ operations. Then, using Montgomery's formulas for scalar multiplication [40], we can compute the points P and $[\ell]P$ without the knowledge of the ordinate of Q, at a cost of $O(r\mathsf{M}(r)\log(q))$ operations. A valid point is obtained after $O(1)$ tries on average.

The computation of the elliptic period α requires $O(\ell/r)$ scalar multiplications by an integer less than ℓ, for a total cost of $O((\mathsf{M}(r)(\ell/r)\log(\ell))$.

Finally, testing that α generates k is done by computing its minimal polynomial, at a cost of $O(r^{(\omega+1)/2})$ operations in $\mathbb{F}_q$ using [51]. □

6. Algorithm selection

The algorithms presented in the previous sections have very similar complexities, and no one stands out as absolute winner. The complexity of all algorithms depends in a non-trivial way on the parameters q and r, and, for Rains' algorithms, on the search for a parameter ℓ and an associated elliptic curve.

This section studies the complexity of the embedding description problem from a global perspective: we explain how to find parameters for Rains' algorithms and criteria to choose the best among the embedding algorithms.

Given parameters $q = p^d$ and r, Rains' cyclotomic algorithm asks for a *small* parameter ℓ such that:

(1) $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle q \rangle \times S$ for some S,
(2) $\langle q \rangle = rs$ for some integer s,
(3) $\gcd(\varphi(\ell), d) = 1$ (see Note 18).

Since r is a prime power, the second condition lets us take a prime power for ℓ too. Indeed if $\mathbb{Z}/\ell\mathbb{Z} \simeq \mathbb{Z}/\ell_1\mathbb{Z} \times \mathbb{Z}/\ell_2\mathbb{Z}$, then either $q \bmod \ell_1$ or $q \bmod \ell_2$ has order a multiple of r. Furthermore, if $\gcd(\ell, r) = 1$, then we can take ℓ prime, since higher powers would not help satisfy the conditions. On the other hand, if $\gcd(\ell, r) \neq 1$, then the algorithms of Section 3 have much better complexity. Hence we shall take ℓ prime.

Given the above constraints, we can rewrite the conditions as:

(1) $\ell = rsv + 1$ for some s, v such that $\gcd(rs, v) = 1$,
(2) $\operatorname{ord}_\ell(q) = rs$,
(3) $\gcd(rsv, d) = 1$.

Remark. Rains remarked that, when $q = 2$ and r is a power of 2 greater than 4, no ℓ can satisfy these constraints because 2 is a quadratic residue modulo any prime of the form $8u + 1$. This case, however, is covered by the Artin–Schreier technique in Section 3.2, we thus ignore it.

In the elliptic algorithm we look for an integer ℓ and a curve $E/\mathbb{F}_q$ that satisfy the preconditions of Algorithm 6, i.e., such that

(1) the Frobenius endomorphism π satisfies a characteristic equation
$$(\pi - \lambda)(\pi - \mu) = 0 \mod \ell,$$
(2) $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle \lambda \rangle \times S$ for some S,
(3) $\#\langle \lambda \rangle = r$, and
(4) $\mu^r \neq 1 \mod \ell$.

As before, we only need to look at prime ℓ. Because $\mu = q/\lambda$, the last condition is equivalent to $q^r \neq 1 \bmod \ell$. Hence, we can restate the conditions on ℓ as

(1) $\ell = ru + 1$ for some u such that $\gcd(r, u) = 1$,
(2) $q^r \neq 1 \mod \ell$.

Once ℓ is found, we compile a list of all integers of order r in $(\mathbb{Z}/\ell\mathbb{Z})^\times$, and look for a curve of trace $t = \lambda + q/\lambda \bmod \ell$ for any λ in the list. Note, however, that for there to be such a curve, t must have a representative in the interval $[-2\sqrt{q}, 2\sqrt{q}]$.

We thus have a procedure to produce parameters for Rains' algorithms: test integers of the form $\ell = ur + 1$ for increasing u, until a suitable one is found. We summarize the elliptic case in Algorithm 7.

Algorithm 7. Parameter selection for Rains' elliptic algorithm

```
Input: Prime powers $q$ and $r$, a bound $\bar{u} \in o(\sqrt[4]{q}/r)$;
Output: An elliptic curve suitable for Rains' elliptic algorithm, or FAIL.
for $u = 1$ to $\bar{u}$ do
    if $\gcd(u, r) = 1$, and $\ell = ur + 1$ is prime, and $q^r \neq 1 \bmod \ell$ then
        Compute $\mathcal{T} \leftarrow \{\lambda + q/\lambda \bmod \ell \mid \mathrm{ord}_\ell(\lambda) = r\}$;
        for each curve $E$ defined over $\mathbb{F}_q$ do
            Compute $t = \#E - q + 1$;
            if $(t \bmod \ell) \in \mathcal{T}$ return $(E, t)$;
        end for
    end if
end for
return FAIL.
```

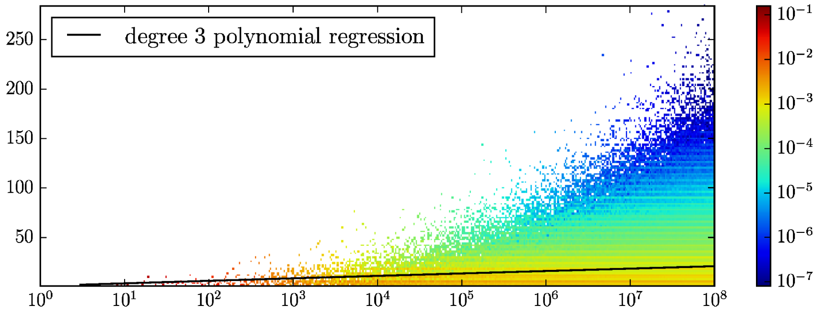


FIGURE 1. Prime powers r (abscissa) versus smallest integer u (ordinate) such that $ur+1$ is prime. Abscissa in logarithmic scale, density normalized by $\log(x)/x$ and colored in logarithmic scale.

Nevertheless, we are left with a question: when does the search for a suitable prime ℓ stop? It is not easy to give a precise answer: the condition that $\ell = ur+1$ is prime already poses some difficulties. Heuristically, we expect that about $u/\log(u)$ of those numbers are prime. However, the best lower bound on primes of the form $\ell = ur + 1$, even under GRH, is $\ell \in O(r^{2.4+\epsilon})$ [21]. Empirical data show that the reality is much closer to the heuristic bound: in Figure 1 we plot for all prime powers $r < 10^8$ the smallest u such that $ur + 1$ is prime. It appears that u is effectively bounded by $O(\log(r))$ for any practical purpose.

For the cyclotomic algorithm we also require that $\mathrm{ord}_\ell(q)$ is a multiple of r. Assuming that q is uniformly distributed in $(\mathbb{Z}/\ell\mathbb{Z})^\times$, this happens with probability at least $\varphi(r)/r \geq 1/2$, hence we can assume that asymptotically $\ell \in O(r\log(r))$.

The elliptic algorithm has a similar condition on ℓ, however, it also requires that the set of curves with trace in $\mathcal{T}$ is not empty. In principle, it is enough to ask that $\ell < 4\sqrt{q}$; however, in order to have a good chance of finding such curves, we set an even more stringent bound $\ell \in o(\sqrt[4]{q})$. Indeed, although it is well known that traces are not evenly distributed modulo prime numbers [35], it is shown in [10, Th. 1] that the probability that the trace of a random curve is in $\mathcal{T}$ approaches $|\mathcal{T}|/\ell \sim r/\ell$, as ℓ and q go to infinity, subject to $\ell \in o(\sqrt[4]{q})$. Thus, also in the elliptic case, we can assume that $\ell \in O(r\log(r))$.

Finally, we must also take into account the possibility that the elliptic algorithm fails on the curve output by Algorithm 7. Under the heuristics about the random distribution of elliptic periods, this possibility only discards one in $O(q^r)$ curves, and is thus negligible.

The cost of the search for ℓ is negligible compared to the cost of computing the embedding: for each candidate $\ell = ur + 1$ we need to test its primality, and do some computations modulo ℓ. All this is well within $O(\sqrt{r})$ binary operations, using naive algorithms. In the elliptic case we also need to count the number of points of $O(\log(r))$ elliptic curves defined over $\mathbb{F}_q$, which can be done in $\tilde{O}\left(\log^5(q)\right)$ binary operations using the Schoof–Elkies–Atkin algorithm [37, 49]. This highlights the fact that the elliptic algorithm is only practical for relatively small q.

Summarizing, we can expect heuristically to find a $\ell \in O(r\log(r))$ that satisfies all the constraints for the cyclotomic algorithm, leading to an expected running time of $\tilde{O}(r^{(\omega+1)/2} + \mathsf{M}(r)\log(q))$ operations in $\mathbb{F}_q$. Similarly, if we assume that $r\log(r) \in o(\sqrt[4]{q})$, we can expect to find suitable parameters for the elliptic algorithm, leading to an expected running time of $\tilde{O}(r^2\log(q))$ operations in $\mathbb{F}_q$, plus $\tilde{O}(\log^5(q))$ binary operations.

Although the complexity of the cyclotomic algorithm looks better, it must not be neglected that the $\tilde{O}$ notation hides the cost of taking an auxiliary extension of degree $O(\log(r))$; whereas the elliptic algorithm, when it applies, does not incur such overhead. The impact of the hidden terms in the complexity can be extremely important, as we will show in the next section.

The same considerations also apply when comparing Rains' algorithms to Allombert's. Indeed, the latter performs extremely well when the degree s of the auxiliary extension is small, but becomes slower as this degree increases.

In practice, it is hopeless to try and determine the appropriate bounds for each algorithm from a purely theoretical point of view. The best approach we can suggest, is to determine parameters at runtime, and set bounds and thresholds experimentally. To summarize, given parameters q and r, we suggest the following approach:

1. If $\gcd(q, r) \neq 1$, run the Artin–Schreier algorithm of Section 3.2.
2. If r is a power of a small prime v, run the algorithm of Section 3.3.
3. Determine the order s of q in $(\mathbb{Z}/r\mathbb{Z})^\times$. If it is small enough, run one of the variants of Allombert's algorithm presented in Section 3.
4. Search for suitable parameters for Rains' algorithms. Depending on the best parameters found, run the best option among Rains' cyclotomic algorithm, Rains' elliptic algorithm, and Allombert's algorithm.

In the next section we shall focus on the last two steps, by comparing our implementations of the algorithms involved, thus giving an estimate of the various thresholds between them. However, we stress that these thresholds are bound to vary depending on the implementation and the target platform; thus it is the implementer's responsibility to determine them at the moment of configuring the system.

7. Experimental results

To validate our results, we implemented the algorithms described in the previous sections, and compared them to the implementation of Allombert's algorithm available in PARI/GP 2.8.0 [54], and to that of Rains' algorithm available in Magma

2.22-7 [4]. The variants of Allombert's algorithm described in Section 3.1 were implemented in C on top of the Flint library, version 2.5.2 [19]. Rains' cyclotomic and elliptic algorithms were implemented in SageMath, version 7.5.rc0 [15] (which itself uses PARI and Flint to implement finite fields), with critical code rewritten in C/Cython. Our code is limited to q a prime fitting in a machine-word, and m, n odd.

We make our code freely available under the open source MIT license, at `https://github.com/defeo/ffisom`. Instructions for compiling and running it are available online, as well as a ready-made cloud environment for testing it, hosted by the Binder service. We stress the fact that our code is experimental and only meant for research purposes; a subset of the authors is currently working on integrating part of it in a future release of Flint.

We ran tests for a wide range of primes q between 3 and $2^{60}+253$, and prime powers r between 3 and 2069. All tests were run on an Intel(R) Xeon(R) CPU E5-4650 v2 clocked at 2.40GHz. We report in Figure 2 some statistics, exclusively on the runs for $100 < q < 2^{20}$; other ranges show very similar trends. The complete datasets, together with more plots and interactive visualizations are also hosted at `https://github.com/defeo/ffisom`.

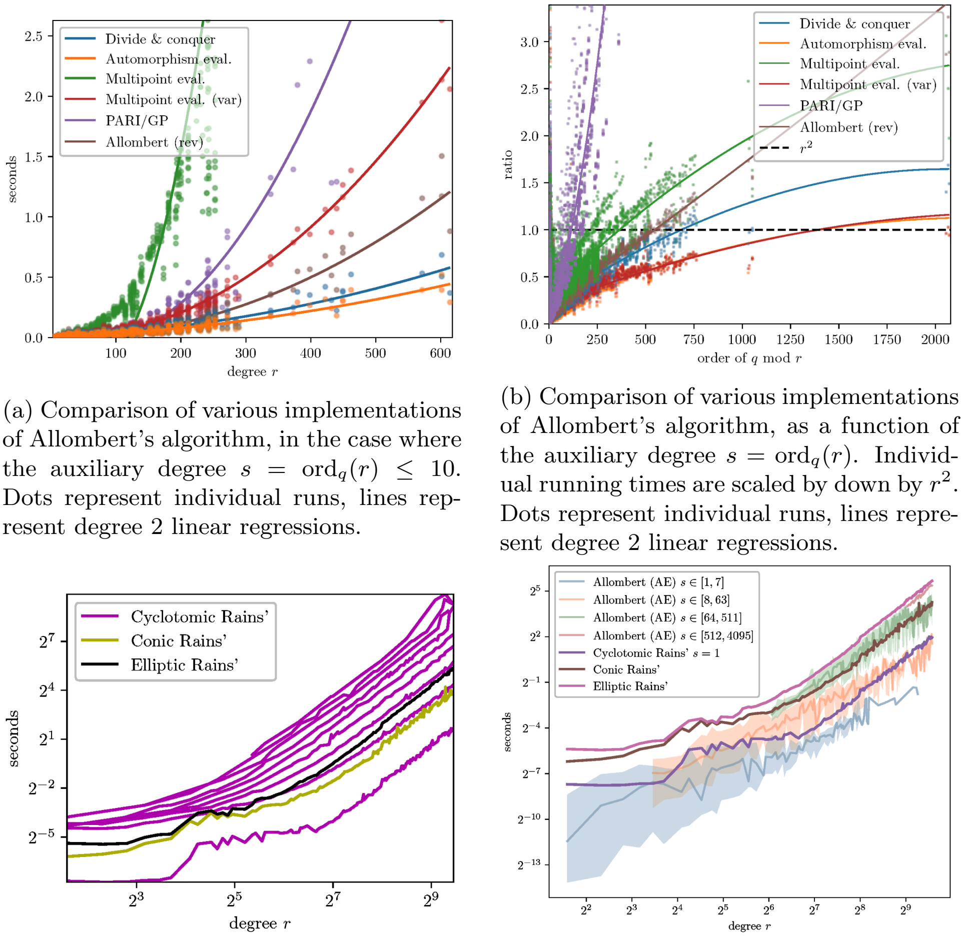
We start by comparing our implementation of the three variants of Allombert's algorithm presented in Section 3.1.3 with the original one in PARI. In Figure 2a we plot running times against the extension degree r, only for cases where the auxiliary degree $s = \mathrm{ord}_q(r)$ is at most 10: dots represent individual runs, continuous lines represent degree 2 linear regressions. Analyzing the behavior for arbitrary auxiliary degree s is more challenging. Based on the observation that all variants have essentially quadratic cost in r, in Figure 2b we take running times, we scale them down by r^2, and we plot them against the auxiliary degree s.

The first striking observation is the extremely poor performance of PARI, especially as s grows. To provide a fairer comparison, we reimplemented Allombert's revised algorithm [3], as faithfully as possible, as described in Section 3.1.2; this is the curve labeled *"Allombert (rev)"* in the graphs. For completeness we also implemented the Paterson–Stockmeyer variant described previously; we do not plot it here, because it overlaps almost perfectly with our *"Divide & conquer"* curve. Although our reimplementations are considerably faster than PARI, it is apparent that Allombert's original algorithm does not behave as well as our new variants.

Focusing now on our three new variants presented in Section 3.1.3, one can't fail to notice that the second one, named *"Automorphism evaluation"*, beats the other two by a great margin, both for small and large auxiliary degree. Although the *"Multipoint evaluation"* approach is expected to eventually beat the other variants as s grows, the cross point seems to be extremely far from the parameters we explored. However, we notice that the naive variant of *"Multipoint evaluation"* not using the iterated Frobenius technique (labeled *"Multipoint evaluation (var)"* in the graphs), starts poorly, then quickly catches *"Automorphism evaluation"* as s grows.

Finally, comparing the variants as q grows shows[7] that the hierarchy between them is essentially unchanged as q approaches 2^{64}. We could not test larger values of q, as our code is limited to machine-word size.

[7] For lack of space we do not show these graphs here, but see `https://github.com/defeo/ffisom/blob/master/notebooks/benchmarks.ipynb`.



(a) Comparison of various implementations of Allombert's algorithm, in the case where the auxiliary degree $s = \mathrm{ord}_q(r) \leq 10$. Dots represent individual runs, lines represent degree 2 linear regressions.

(b) Comparison of various implementations of Allombert's algorithm, as a function of the auxiliary degree $s = \mathrm{ord}_q(r)$. Individual running times are scaled by down by r^2. Dots represent individual runs, lines represent degree 2 linear regressions.

(c) Cyclotomic, conic and elliptic variants of Rains' algorithm. Auxiliary extension degrees s for cyclotomic Rains' range between 1 and 9. Lines represent median times.

(d) Comparison of Allombert's (Automorphism evaluation variant) and Rains' algorithms at some fixed auxiliary extension degrees s. Lines represent median times, shaded areas minimum and maximum times.

FIGURE 2. Benchmarks for Rains' and Allombert's algorithms. q is a prime between 100 and 2^{20}, r is an odd prime power varying between 3 and 2069. Plots c and d are in doubly logarithmic scale.

Now we shift to Rains' algorithm and its variants. In comparing our implementation with Magma's, discarding outliers, we obtain a fairly consistent speed-up of about 30% (see Figure 3); hence we will compare these algorithms only based on our timings. In Figure 2c we group runs of the cyclotomic algorithm by the degree s of the auxiliary extension, and we plot median times against the degree r; only the graphs for $s < 10$ are shown in the figure. We observe a very large gap between $s = 1$ and larger s ($s = 2$ is $8-16$ times slower). This is partly due to the fact that we use generic Python code to construct auxiliary extensions, rather than dedicated

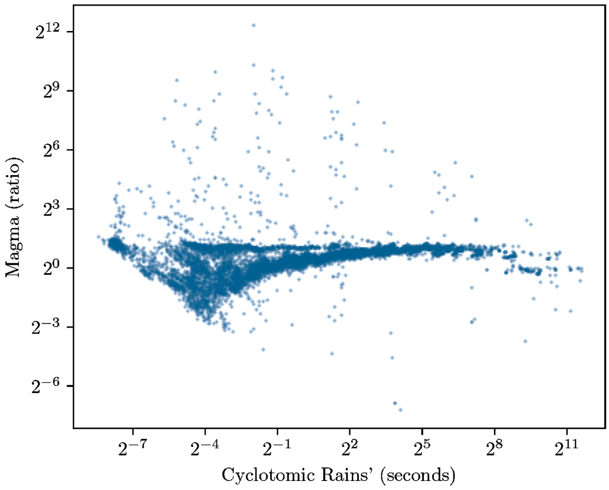


FIGURE 3. Comparison of our implementation of Rains' algorithm and Magma's. Running time of our implementation in seconds vs. ratio of Magma running time over ours. Plot in doubly logarithmic scale.

C; however, a large gap is unavoidable, due to the added cost of computing in extension fields. We also plot median times for the elliptic variant and for the conic variant (see Appendix A). It is apparent that the elliptic algorithm outperforms the cyclotomic one as soon as $s \geq 3$, and that the conic algorithm conveniently replaces the case $s = 2$. Thus, at least for the parameter ranges we have tested, the cyclotomic algorithm with auxiliary extensions seems of limited interest. Plotting against the prime q instead of the degree r shows essentially the same behavior[8], however, it should be reminded that the elliptic variant becomes very poor when q grows larger than a machine-word, because of the cost of point counting.

Finally, in Figure 2d we compare Rains' algorithms against Allombert's. In light of the excellent performances of the *"Automorphism evaluation"* variant of Allombert's algorithm, we only plot the performances for this variant. We plot, against the degree r, runs of Allombert's algorithm grouped by ranges of the auxiliary degree $\operatorname{ord}_r(q)$: we shade the area between minimum and maximum running times, and trace the median time. We also take from Figure 2c the graphs for the cyclotomic (only $s = 1$), the conic and the elliptic variants of Rains' algorithm. We notice that Allombert's algorithm, even with relatively large auxiliary degrees, is extremely fast; the cyclotomic algorithm only beats it when $\operatorname{ord}_r(q)$ goes beyond 10 to 50, the conic algorithm only beats extremely large $\operatorname{ord}_r(q)$, and the elliptic algorithm is never better. We also observe that Allombert's algorithm has a better asymptotic behavior as the degree r grows.

In light of these comparisons, it seems that the absolute winner is our *Automorphism evaluation* variant of Allombert's algorithm, with Rains' cyclotomic algorithm being only occasionally more interesting. Obviously, the comparisons are

[8] Again, see `https://github.com/defeo/ffisom/blob/master/notebooks/benchmarks.ipynb` for a plot.

Algorithm 8. Rains' conic algorithm

```
Input: A field extension $k/\mathbb{F}_q$ of degree $r$; a prime $\ell$ such that
    • $(\mathbb{Z}/\ell\mathbb{Z})^\times = \langle q \rangle \times S$ for some $S$,
    • $\#\langle q \rangle = 2rs$ for some integer $s$;
  a polynomial $h$ of degree $s$ irreducible over $k$.
Output: A normal generator of $k$ over $\mathbb{F}_q$, with a uniquely defined Galois orbit.
  Construct the field extension $k' = k[Z]/h(Z)$;
  repeat
    repeat
      Take a random element $\theta \in k'$,
    until $\theta^2 - 4$ is a quadratic non-residue;
    Compute $\zeta = [(\#k' + 1)/\ell]\theta$,
  until $\zeta \neq 2$;
  Compute $\eta(\zeta) \leftarrow \sum_{\sigma\in S}[\sigma]\zeta$;
  return  $\alpha \leftarrow \mathrm{Tr}_{k'/k}\,\eta(\zeta) = \sum_{i=0}^{s-1}[q^{ri}]\eta(\zeta)$.
```

only relevant to our own code and test conditions. Other implementations and benchmarks will likely find slightly different cross-points for the algorithms.

Appendix A. Rain's conic algorithm

We have seen that Rains' cyclotomic algorithm suffers in practice from the need to build a field extension k' of k. The conic variant we are going to present reduces the degree of the field extension from $s = [k' : k]$ to $s/2$ whenever s is even. This is especially useful when $s = 2$, as highlighted in Section 7. The algorithm is similar in spirit to Williams' $p+1$ factoring method [57], where the arithmetic of the norm 1 subgroup of k'^* is performed using Lucas sequences on a subfield of index 2 of k'.

Let $\mathbb{F}$ be a finite field of odd characteristic, let $\Delta \in \mathbb{F}$ be a quadratic non-residue, let δ be an element of the algebraic closure of $\mathbb{F}$ such that $\delta^2 = \Delta$, and define the norm 1 subgroup of $\mathbb{F}[\delta]^*$ as

$$T_2(\mathbb{F}) = \{(x + \delta y)/2 \mid x, y \in \mathbb{F} \text{ and } x^2 - \Delta y^2 = 4\};$$

it is easy to verify that $T_2(\mathbb{F})$ forms a group under multiplication. If we see the elements $(x + \delta y)/2$ as points (x, y) on a conic $x^2 - \Delta y^2 = 4$, the group law of $T_2(\mathbb{F})$ induces a group law on the conic. By projecting onto the x-coordinate, a straightforward calculation shows that, for any point $(\theta, *)$ on the conic, its nth power has coordinates $(\theta_n, *)$, where θ_n is defined by the Lucas sequence

$$\theta_0 = 2, \quad \theta_1 = \theta, \quad \theta_{i+1} = \theta\theta_i - \theta_{i-1}.$$

We shall denote by $[n]$ the map $\theta \mapsto \theta_n$; notice how it does not depend on the choice of Δ.

The generalization of Rains' algorithm is now obvious: by projecting on the x-coordinate, we work in a field extension twice as small compared to the original algorithm. This is summarized in Algorithm 8.

Proposition 25. *Algorithm* 8 *is correct: on input* q, r, ℓ, s *it returns an element in the same Galois orbit as Algorithm* 5 *on input* $q, r, \ell, 2s$. *It computes its output using* $O(\mathsf{M}(sr)(sr\log(q) + (\ell/r)\log(\ell)))$ *operations in* $\mathbb{F}_q$ *on average, or* $\tilde{O}((sr)^2\log(q))$ *assuming* $\ell \in o(sr^2)$.

Proof. By construction, all the ℓth roots of unity are in $T_2(k')$. Observe that if $(x+\delta y)/2$ is in $T_2(k')$, then its trace over k' is equal to x. Hence, the value ζ computed in Step 6 is the trace over k' of a primitive ℓth root of unity. We conclude by comparing this algorithm with Algorithm 5.

The non-residuosity test in Step 5 is done by verifying that the $(\#k'-1)/2$th power of θ is equal to -1. We do this in $O(sr\log(q))$ operations in k', or $O(sr\mathsf{M}(sr)\cdot\log(q))$ operations in $\mathbb{F}_q$.

To implement the other steps, we need to evaluate the map $[n]$ efficiently. We have the following classical relationships for the Lucas sequence of θ:

$$\theta_{2i}=\theta_i^2-2,\quad \theta_{2i+1}=\theta_i\theta_{i+1}-\theta,\quad \theta_{2i+2}=\theta_{i+1}^2-2.$$

Starting with $\theta_0=2$ and $\theta_1=\theta$, we use a binary scheme to deduce θ_i,θ_{i+1} from $\theta_{\lfloor i/2\rfloor},\theta_{\lfloor i/2\rfloor+1}$. We reach θ_n after $O(\log(n))$ steps, each requiring a constant number of operations in k'.

Hence, Step 6 costs $O(sr\mathsf{M}(sr)\log(q))$ operations in $\mathbb{F}_q$, while Steps 8 and 9 together cost $O((\mathsf{M}(sr)(\ell/r)\log(\ell))$. □

Although this variant does not exploit the asymptotic improvement offered by Proposition 2, the fact that its auxiliary degree s is half the one of the original algorithm usually gives an interesting practical improvement. Step 6 can be modified so as to avoid the premature projection on the x-axis, so that the algorithms of Proposition 2 apply. We leave the details of this variant to the reader.

Acknowledgments

We would like to thank Eric M. Rains for sharing his preprint with us. We also thank Bill Allombert, Christian Berghoff, Jean-Marc Couveignes, Reynald Lercier, and Benjamin Smith for fruitful discussions.

References

[1] L. M. Adleman and H. W. Lenstra. Finding irreducible polynomials over finite fields. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 350–355, New York, NY, USA, 1986. ACM.

[2] B. Allombert, *Explicit computation of isomorphisms between finite fields*, Finite Fields Appl. **8** (2002), no. 3, 332–342. MR1910396

[3] Bill Allombert, *Explicit computation of isomorphisms between finite fields*, Revised version. `https://www.math.u-bordeaux.fr/~ballombe/fpisom.ps`, 2002.

[4] W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265. Computational Algebra and Number Theory (London, 1993). MR1484478

[5] W. Bosma, J. Cannon, and A. Steel, *Lattices of compatibly embedded finite fields*, J. Symbolic Comput. **24** (1997), no. 3-4, 351–369. Computational algebra and number theory (London, 1993). MR1484485

[6] R. P. Brent and H. T. Kung, *Fast algorithms for manipulating formal power series*, J. Assoc. Comput. Mach. **25** (1978), no. 4, 581–595. MR0520733

[7] Ludovic Brieulle, Luca De Feo, Javad Doliskani, Jean-Pierre Flori, and Éric Schost, *Computing isomorphisms and embeddings of finite fields* (extended version), arXiv preprint arXiv:1705.01221, 2017.

[8] D. G. Cantor and E. Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Inform. **28** (1991), no. 7, 693–701. MR1129288

[9] D. G. Cantor and H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. Comp. **36** (1981), no. 154, 587–592. MR606517

[10] W. Castryck and H. Hubrechts, *The distribution of the number of points modulo an integer on elliptic curves over finite fields*, Ramanujan J. **30** (2013), no. 2, 223–242. MR3017927

[11] J.-M. Couveignes and R. Lercier, *Galois invariant smoothness basis*, Algebraic geometry and its applications, Ser. Number Theory Appl., vol. 5, World Sci. Publ., Hackensack, NJ, 2008, pp. 142–167. MR2484053

[12] J.-M. Couveignes and R. Lercier, *Fast construction of irreducible polynomials over finite fields*, Israel J. Math. **194** (2013), no. 1, 77–105. MR3047063

[13] L. De Feo, J. Doliskani, and É. Schost, *Fast algorithms for ℓ-adic towers over finite fields*, ISSAC 2013—Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2013, pp. 165–172. MR3206354

[14] L. De Feo, J. Doliskani, and É. Schost, *Fast arithmetic for the algebraic closure of finite fields*, ISSAC 2014—Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2014, pp. 122–129. MR3239917

[15] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.5.rc0)*, 2016. http://www.sagemath.org.

[16] J. Doliskani and É. Schost, *Taking roots over high extensions of finite fields*, Math. Comp. **83** (2014), no. 285, 435–446. MR3120598

[17] S. Feisel, J. von zur Gathen, and M. A. Shokrollahi, *Normal bases via general Gauss periods*, Math. Comp. **68** (1999), no. 225, 271–290. MR1484903

[18] C. F. Gauss, *Disquisitiones arithmeticae*, Translated into English by Arthur A. Clarke, S. J, Yale University Press, New Haven, Conn.-London, 1966. MR0197380

[19] W. B. Hart, *Fast library for number theory: an introduction*, Mathematical software—ICMS 2010, Lecture Notes in Comput. Sci., vol. 6327, Springer, Berlin, 2010, pp. 88–91. MR3663175

[20] D. Harvey, *Faster polynomial multiplication via multipoint Kronecker substitution*, J. Symbolic Comput. **44** (2009), no. 10, 1502–1510. MR2543433

[21] D. R. Heath-Brown, *Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression*, Proc. London Math. Soc. (3) **64** (1992), no. 2, 265–338. MR1143227

[22] E. Kaltofen, *Computer algebra algorithms*, Annual review of computer science, Vol. 2, Annual Reviews, Palo Alto, CA, 1987, pp. 91–118. MR921493

[23] E. Kaltofen and V. Shoup, *Fast polynomial factorization over high algebraic extensions of finite fields*, Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (Kihei, HI), ACM, New York, 1997, pp. 184–188. MR1809986

[24] E. Kaltofen and V. Shoup, *Subquadratic-time factoring of polynomials over finite fields*, Math. Comp. **67** (1998), no. 223, 1179–1197. MR1459389

[25] K. S. Kedlaya and C. Umans, *Fast polynomial factorization and modular composition*, SIAM J. Comput. **40** (2011), no. 6, 1767–1802. MR2863194

[26] E. E. Kummer, *Mémoire sur la théorie des nombres complexes composés de racines de l'unité et de nombres entiers*, Journal de Mathématiques Pures et Appliquées, (1851), 377–498.

[27] E. E. Kummer, *Über die Divisoren gewisser Formen der Zahlen, welche aus der Theorie der Kreistheilung entstehen* (German), J. Reine Angew. Math. **30** (1846), 107–116. MR1578458

[28] E. E. Kummer, *Sur les nombres complexes qui sont formés avec les nombres entiers réels et les racines de l'unité*, Journal de Mathématiques Pures et Appliquées, (1847), 185–212.

[29] E. E. Kummer, *Über die Zerlegung der aus Wurzeln der Einheit gebildeten complexen Zahlen in ihre Primfactoren* (German), J. Reine Angew. Math. **35** (1847), 327–367. MR1578599

[30] E. E. Kummer, *Zur Theorie der complexen Zahlen* (German), J. Reine Angew. Math. **35** (1847), 319–326. MR1578598

[31] E. E. Kummer, *Über eine besondere Art, aus complexen Einheiten gebildeter Ausdrücke* (German), J. Reine Angew. Math. **50** (1855), 212–232. MR1578935

[32] E. E. Kummer, *Über die den Gaußschen Perioden der Kreistheilung entsprechenden Congruenzwurzeln* (German), J. Reine Angew. Math. **53** (1857), 142–148. MR1578992

[33] S. Lang, *Algebra*, 3rd ed., Graduate Texts in Mathematics, vol. 211, Springer-Verlag, New York, 2002. MR1878556

[34] F. Le Gall, *Powers of tensors and fast matrix multiplication*, ISSAC 2014—Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2014, pp. 296–303. MR3239939

[35] H. W. Lenstra Jr., *Factoring integers with elliptic curves*, Ann. of Math. (2) **126** (1987), no. 3, 649–673. MR916721

[36] H. W. Lenstra Jr., *Finding isomorphisms between finite fields*, Math. Comp. **56** (1991), no. 193, 329–347. MR1052099

[37] R. Lercier and T. Sirvent, *On Elkies subgroups of l-torsion points in elliptic curves defined over a finite field*, J. Théor. Nombres Bordeaux **20** (2008), no. 3, 783–797. MR2523317

[38] P. Mihăilescu and V. Vuletescu, *Elliptic Gauss sums and applications to point counting*, J. Symbolic Comput. **45** (2010), no. 8, 825–836. MR2657666

[39] R. T. Moenck, *Another polynomial homomorphism*, Acta Informat. **6** (1976), no. 2, 153–169. MR0408306

[40] P. L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Math. Comp. **48** (1987), no. 177, 243–264. MR866113

[41] G. L. Mullen (ed.), *Handbook of Finite Fields*, Discrete Mathematics and its Applications (Boca Raton), CRC Press, Boca Raton, FL, 2013. MR3087321

[42] A. K. Narayanan, *Fast computation of isomorphisms between finite fields using elliptic curves*, arXiv preprint arXiv:1604.03072, 2016.

[43] E. Noether, *Normalbasis bei Körpern ohne höhere Verzweigung* (German), J. Reine Angew. Math. **167** (1932), 147–152. MR1581331

[44] C. Pascal and É. Schost, *Change of order for bivariate triangular sets*, ISSAC 2006, ACM, New York, 2006, pp. 277–284. MR2289131

[45] M. S. Paterson and L. J. Stockmeyer, *On the number of nonscalar multiplications necessary to evaluate polynomials*, SIAM J. Comput. **2** (1973), 60–66. MR0314238

[46] R. G. E. Pinch, *Recognising elements of finite fields*, Cryptography and coding, II (Cirencester, 1989), Inst. Math. Appl. Conf. Ser. New Ser., vol. 33, Oxford Univ. Press, New York, 1992, pp. 193–197. MR1165738

[47] A. Poteaux and É. Schost, *Modular composition modulo triangular sets and applications*, Comput. Complexity **22** (2013), no. 3, 463–516. MR3090784

[48] E. M. Rains, *Efficient computation of isomorphisms between finite fields*, personal communication, 1996.

[49] R. Schoof, *Counting points on elliptic curves over finite fields*, J. Théor. Nombres Bordeaux **7** (1995), no. 1, 219–254. Les Dix-huitièmes Journées Arithmétiques (Bordeaux, 1993). MR1413578

[50] V. Shoup, *New algorithms for finding irreducible polynomials over finite fields*, Math. Comp. **54** (1990), no. 189, 435–447. MR993933

[51] V. Shoup, *Fast construction of irreducible polynomials over finite fields*, Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993), ACM, New York, 1993, pp. 484–492. MR1213261

[52] V. Shoup, *Fast construction of irreducible polynomials over finite fields*, J. Symbolic Comput. **17** (1994), no. 5, 371–391. MR1289997

[53] V. Shoup, *Efficient computation of minimal polynomials in algebraic extensions of finite fields*, Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation (Vancouver, BC), ACM, New York, 1999, pp. 53–58. MR1802067

[54] The PARI Group, Bordeaux. *PARI/GP, version* `2.8.0`, 2016.

[55] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, New York, 1999. MR1689167

[56] J. von zur Gathen and V. Shoup, *Computing Frobenius maps and factoring polynomials*, Comput. Complexity **2** (1992), no. 3, 187–224. MR1220071

[57] H. C. Williams, *A $p+1$ method of factoring*, Math. Comp. **39** (1982), no. 159, 225–234. MR658227

Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay
Email address: l.brieulle@gmail.com

Laboratoire de Mathématiques de Versailles, UVSQ, CNRS & Inria, Université Paris-Saclay
Email address: luca.de-feo@uvsq.fr
URL: http://orcid.org/0000-0002-9321-0773

Institute for Quantum Computing, University of Waterloo
Email address: javad.doliskani@uwaterloo.ca

Agence nationale de la sécurité des systèmes d'information
Email address: jean-pierre.flori@ssi.gouv.fr

Cheriton School of Computer Science, University of Waterloo
Email address: eschost@uwaterloo.ca