Drinfeld Modules with Complex Multiplication, Hasse Invariants and Factoring Polynomials over Finite Fields

Javad Doliskani* Anand Kumar Narayanan† Éric Schost‡

August 26, 2018

Abstract

We present a novel randomized algorithm to factor polynomials over a finite field $\mathbb{F}_q$ of odd characteristic using rank 2 Drinfeld modules with complex multiplication. The main idea is to compute a lift of the Hasse invariant (modulo the polynomial $f \in \mathbb{F}_q[x]$ to be factored) with respect to a random Drinfeld module ϕ with complex multiplication. Factors of f supported on prime ideals with supersingular reduction at ϕ have vanishing Hasse invariant and can be separated from the rest. Incorporating a Drinfeld module analogue of Deligne's congruence, we devise an algorithm to compute the Hasse invariant lift, which turns out to be the crux of our algorithm. The resulting expected runtime of $n^{3/2+\varepsilon}(\log q)^{1+o(1)} + n^{1+\varepsilon}(\log q)^{2+o(1)}$ to factor polynomials of degree n over $\mathbb{F}_q$ matches the fastest previously known algorithm, the Kedlaya-Umans implementation of the Kaltofen-Shoup algorithm.

Keywords Elliptic modules, Drinfeld modules, Polynomial factorization, Hasse invariant, Complex multiplication

1 Introduction

Drinfeld modules of rank two are often presented as an analogue over function fields such as $\mathbb{F}_q(x)$ (for a prime power q) of elliptic curves over $\mathbb{Q}$; following Drinfeld's original terminology, we will often call them *elliptic modules* in this paper. In very concrete terms, a Drinfeld module over $\mathbb{F}_q(x)$ is simply a ring homomorphism ϕ (together with some mild assumptions) from $\mathbb{F}_q[x]$ to the ring of skew polynomials $\mathbb{F}_q(x)\{\tau\}$, where τ satisfies the commutation relation $\tau u = u^q \tau$ for u in $\mathbb{F}_q(x)$. The rank of a Drinfeld module is the degree of ϕ_x, where as customary, we write $\phi_a := \phi(a)$ for a in $\mathbb{F}_q[x]$.

In this definition, one may replace $\mathbb{F}_q(x)$ by any other other field L equipped with a homomorphism $\mathbb{F}_q[x] \to L$, and in particular by a finite field of the form $L = \mathbb{F}_q[x]/f$; one may then define the reduction of a Drinfeld module over $\mathbb{F}_q(x)$ modulo an irreducible $f \in \mathbb{F}_q[x]$. Then, there exist striking similarities between the theory of elliptic curves over $\mathbb{Q}$ and their reductions modulo primes, and that of elliptic modules over $\mathbb{F}_q(x)$ and their reduction modulo irreducible polynomials f. For instance, such notions as endomorphism ring, complex multiplication, Hasse invariants, supersingularity, or the characteristic polynomial of the Frobenius, ... can be defined in both contexts, and share many properties. On the other hand, while the literature on algorithmic aspects

*Institute for Quantum Computing, University of Waterloo (javad.doliskani@uwaterloo.ca).
†Laboratoire d'Informatique de Paris 6, Pierre et Marie Curie University (Anand.Narayanan@lip6.fr)
‡Cheriton School of Computer Science, University of Waterloo (eschost@uwaterloo.ca)

of elliptic curves is extremely rich, this is not the case for Drinfeld modules; only recently have they been considered under the algorithmic viewpoint (for instance, it is known that they are not suitable for usual forms of public key cryptography [31]).

In this article, we give an algorithm for the computation of the Hasse invariant of elliptic modules over finite fields, and show how efficient algorithms for this particular problem (and a natural generalization thereof) can be used to factor polynomials over finite fields.

To wit, recall that the Hasse invariant h_E of an elliptic curve $E : y^2 = x^3 + Ax + B$ over a finite field $\mathbb{F}_p$, $p > 2$, can be defined as the coefficient of degree $p-1$ in $(x^3 + Ax + B)^{(p-1)/2}$ (other definitions set it to be 1 if this coefficient is nonzero, 0 otherwise). The definition of h_E makes it possible to compute it using a number of operations softly linear in p, but one can do better: it is possible to compute h_E without computing all previous coefficients, using the fact that the coefficients of $(x^3 + Ax + B)^{(p-1)/2}$ satisfy a linear recurrence with polynomial coefficients, and applying techniques for such recurrences due to Strassen [35] and Chudnovsky and Chudnovsky [8] (see also [3]).

In the case of elliptic modules, we will consider $\phi : \mathbb{F}_q[x] \to \mathbb{F}_q(x)\{\tau\}$, such that $\phi(x) = x + g_\phi \tau + \Delta_\phi \tau^2$, with g_ϕ and Δ_ϕ in $\mathbb{F}_q[x]$. If f is irreducible of degree k and does not divide Δ_ϕ, we say that ϕ has *good reduction* at f. Then, the *Hasse invariant* $h_{\phi,f}$ is defined as the coefficient of τ^k in $\phi_f \bmod f = \sum_{i=0}^{2k} h_i \tau^i$, with h_i in $\mathbb{F}_q[x]/f$ for all i (all coefficients of index less than k vanish modulo f). For an arbitrary squarefree f, using a recurrence due to Gekeler [18] (which is somewhat similar to the one used for elliptic curves), it is possible to define a related quantity which we will write $\bar{h}_{\phi,f}$, in a way that ensures arithmetic properties needed for the factorization algorithm below (see Eq. (4) for the definition).

As in the case of elliptic curves, one can deduce a straightforward algorithm from the definition of $\bar{h}_{\phi,f}$; our first contribution is to show that a better algorithm exist. We will consider two different complexity models in our runtime analysis: an *algebraic model* and a *boolean model*. In the former we count the number of operations $+, \times, \div$ in the field $\mathbb{F}_q$, while in the latter we count the number of bit operations, over a standard RAM (the main reason behind this dichotomy is that some operation at the core of our algorithms, namely *modular composition*, admits faster algorithms in the boolean model; we discuss this further in the next section). We denote by M a function such that polynomials of degree n over any ring can be multiplied in $\mathsf{M}(n)$ base ring operations, and such that the superlinearity conditions of [15, Chapter 8] are satisfied; we can take $\mathsf{M}(n) \in O(n \log n \log\log n)$ [6]. We let ω be a feasible exponent for square matrix multiplication; we can always take $\omega \le 2.38$ [10, 26]. Then, our first main result is the following theorem.

Theorem 1.1. *Let $\phi : \mathbb{F}_q[x] \to \mathbb{F}_q(x)\{\tau\}$ be an elliptic module, such that $\phi(x) = x + g_\phi \tau + \Delta_\phi \tau^2$, with g_ϕ and Δ_ϕ in $\mathbb{F}_q[x]$. Given a squarefree polynomial f of degree n, as well as $g_\phi \bmod f$ and $\Delta_\phi \bmod f$, one can compute $\bar{h}_{\phi,f}$ using*

$$O(n^{(1-\beta)(\omega-1)/2+(\omega+1)/2} + \mathsf{M}(n^{1+\beta}) \log qn)$$

operations in $\mathbb{F}_q$, for any $\beta \in (0,1)$, or using

$$O(n^{3/2+\varepsilon} (\log q)^{1+o(1)} + n^{1+\varepsilon} (\log q)^{2+o(1)})$$

bit operations, for any $\varepsilon > 0$.

The algorithm is inspired by the baby steps / giant steps algorithms for recurrences with polynomial coefficients of [35, 8, 3], and also borrows heavily from Kaltofen and Shoup's baby steps / giant steps distinct degree factorization algorithm [23]; indeed, the structures of the algorithms are similar, and the runtime reported here is the same as that in that reference. As in [23], for our first claim, taking $\omega \approx 2.375$ and $\beta \approx 0.815$, the complexity is $O(n^{1.815} \log q)$ operations in $\mathbb{F}_q$.

We use these results to design a polynomial factorization algorithm. The resulting runtime is not better than that in [23] (in the algebraic model) or [24] (in the boolean model); however, we believe it is worth stating such results, since they bring a new perspective to polynomial factorization questions.

The use of Drinfeld modules for polynomial factorization actually goes back to work of Panchishkin and Potemine [29], whose algorithm was rediscovered by van der Heiden [36]. These algorithms, along with the second author's Drinfeld module black box Berlekamp algorithm [28] are in spirit Drinfeld module analogues of Lenstra's elliptic curve method to factor integers [27]. The Drinfeld module degree estimation algorithm of [28] uses Euler-Poincaré characteristics of Drinfeld modules to estimate the factor degrees in distinct degree factorization. A feature common to the aforementioned algorithms is their use of random Drinfeld modules, which typically don't have complex multiplication.

We take a different approach. To factor a squarefree polynomial f, we construct a random elliptic module ϕ with complex multiplication by an imaginary quadratic extension of the rational function field $\mathbb{F}_q(x)$ with class number 1. At roughly half of the prime ideals $\langle g \rangle$ in $\mathbb{F}_q[x]$, ϕ has so-called supersingular reduction. Concretely, the properties of the quantity $\bar{h}_{\phi,f}$ computed modulo f imply that for any prime factor g of f, g is a prime with supersingular reduction for ϕ if and only if $\bar{h}_{\phi,f} = 0 \bmod g$. The construction of ϕ ensures that this happens with non-vanishing probability. Altogether, we obtain the following result.

Theorem 1.2. *Suppose that given any squarefree polynomial f of degree n over $\mathbb{F}_q$, and any Drinfeld module ϕ as in Theorem 1.1, one can compute $\bar{h}_{\phi,f}$ in $H(n,q)$ operations in $\mathbb{F}_q$, resp. $H^*(n,q)$ bit operations. Suppose also that $H(n_1,q) + H(n_2,q) \leq H(n_1+n_2,q)$, resp. $H^*(n_1,q) + H^*(n_2,q) \leq H^*(n_1+n_2,q)$, holds for all $n_1, n_2 \geq 0$. Then there is a randomized algorithm that can factor degree n polynomials over $\mathbb{F}_q$ in an expected*

- $\tilde{O}(H(n,q) + n^{(\omega+1)/2} + \mathsf{M}(n) \log q)$ *operations in $\mathbb{F}_q$, or*
- $\tilde{O}(H^*(n,q)) + n^{1+\varepsilon}(\log q)^{2+o(1)}$ *bit operations.*

The runtime reported in the first item in Theorem 1.1 imply that we can take $H(n,q) = \kappa n^{1.815} \log q$, for a suitable constant κ. Since such a function satisfies the superlinearity assumption of Theorem 1.2, the resulting runtime for the factoring algorithm is is $O(n^{1.815} \log q)$ operations in $\mathbb{F}_q$. In a boolean model, the second item in Theorem 1.1 shows that for any $\varepsilon > 0$, we can take $H^*(n,q) = \kappa^*_\varepsilon (n^{3/2+\varepsilon}(\log q)^{1+o(1)} + n^{1+\varepsilon}(\log q)^{2+o(1)})$ bit operations, for some constant κ^*_ε; superlinearity still holds, which then implies a similar runtime for factorization.

As mentioned above, these results do not improve on the state-of-the-art for polynomial factorization. Since Berlekamp's randomized polynomial time algorithm [1], polynomial factorization over finite fields has been the subject of a vast body of work. Among important milestones, we mention [5, 16]; the first subquadratic algorithms are due to Kaltofen and Shoup [23], with runtimes $O(n^{1.815} \log q)$ operations in $\mathbb{F}_q$ (two algorithms are in that reference: a fast distinct degree factorization algorithm, and an algorithm derived from the Berlekamp algorithm).

Kaltofen and Shoup already pointed out that a quasi-linear time algorithm for modular composition would yield an exponent 3/2 for polynomial factorization; Kedlaya and Umans [24] showed that modular composition can be done in time $n^{1+\varepsilon}(\log q)^{1+o(1)}$ in a boolean model, and exhibited the resulting algorithm for factoring polynomials, with runtime $n^{3/2+\varepsilon}(\log q)^{1+o(1)} + n^{1+\varepsilon}(\log q)^{2+o(1)}$ bit operations.

We remark that our algorithm has the property of not requiring separate distinct degree and equal degree factorization phases. In the algebraic model, Kaltofen and Shoup's black box Berlekamp algorithm has a similar runtime and also bypasses the distinct degree / equal degree factorization stages; however, we are not aware of a version of that algorithm that would feature a runtime of $n^{3/2+\varepsilon}(\log q)^{1+o(1)} + n^{1+\varepsilon}(\log q)^{2+o(1)}$ in the boolean model. As to if the exponent 3/2 in n can be lowered, this remains an outstanding open question.

The paper is organized as follows. In §2, elliptic modules are introduced; we define Hasse invariants and give our algorithm for their computation, proving Theorem 1.1. In §3, we describe our factoring algorithm and prove Theorem 1.2, using a few notions from function field arithmetic.

2 Computing Hasse invariant lifts of elliptic modules

2.1 Basic definitions

Let $\mathbb{F}_q[x]$ denote the polynomial ring in the indeterminate x, for some odd prime power q, and let L be a field equipped with a homomorphism $\gamma : \mathbb{F}_q[x] \to L$; typical examples for us will be $L = \mathbb{F}_q(x)$ and $L = \mathbb{F}_q[x]/f$, for some irreducible polynomial f in $\mathbb{F}_q[x]$. Let us further consider the skew polynomial ring $L\{\tau\}$, where τ satisfies the commutation rule $\forall u \in L, \tau u = u^q \tau$. Given an integer $r > 0$, a Drinfeld module of rank r over L is a ring morphism

$$\begin{array}{rccl} \phi: & \mathbb{F}_q[x] & \longrightarrow & L\{\tau\} \\ & x & \longmapsto & a_0 + a_1\tau + \cdots + a_r\tau^r \end{array} \tag{1}$$

with $a_0 = \gamma(x)$ and $a_r \neq 0$.

An *elliptic module* is a rank-2 Drinfeld module, obtained by setting $r = 2$ in (1). Consider such an elliptic module over $L = \mathbb{F}_q(x)$, and let γ be the inclusion $\mathbb{F}_q[x] \to \mathbb{F}_q(x)$. In such a case, ϕ will be written as

$$\begin{array}{rccl} \phi: & \mathbb{F}_q[x] & \longrightarrow & \mathbb{F}_q(x)\{\tau\} \\ & x & \longmapsto & x + g_\phi\tau + \Delta_\phi\tau^2 \end{array}$$

for some $g_\phi \in \mathbb{F}_q[x]$ and nonzero $\Delta_\phi \in \mathbb{F}_q[x]$. For an irreducible polynomial $f \in \mathbb{F}_q[x]$, if Δ_ϕ is nonzero modulo f, then the reduction ϕ/f of ϕ at f is defined as the elliptic module

$$\begin{array}{rccl} \phi/f: & \mathbb{F}_q[x] & \longrightarrow & (\mathbb{F}_q[x]/f)\{\tau\} \\ & x & \longmapsto & x + (g_\phi \bmod f)\tau + (\Delta_\phi \bmod f)\tau^2; \end{array}$$

we say that ϕ has *good reduction* at f in this case. Then, the image of $a \in \mathbb{F}_q[x]$ under ϕ/f is denoted by $(\phi/f)_a$. Even if Δ_ϕ is zero modulo f, one could still obtain the reduction (ϕ/f) of ϕ at f through minimal models of ϕ [17]; we refrain from addressing this case since our algorithms do not require it.

Let ϕ be as above and let $f \in \mathbb{F}_q[x]$ be an irreducible polynomial not dividing Δ_ϕ. The *Hasse invariant* $h_{\phi,f} \in \mathbb{F}_q[x]/f$ of ϕ at f is the coefficient of $\tau^{\deg(f)}$ in the expansion

$$(\phi/f)_f = \sum_{i=0}^{2\deg(f)} h_i\tau^i \in (\mathbb{F}_q[x]/f)\{\tau\}.$$

The elliptic module ϕ has *supersingular reduction* at f if $h_{\phi,f}$ vanishes [19]; otherwise, we say that it has *ordinary reduction*. If the choice of ϕ is clear from context, we will simply call f supersingular.

Recursively define a sequence $(r_{\phi,k})_{k\in\mathbb{N}}$ in $\mathbb{F}_q[x]^{\mathbb{N}}$ as $r_{\phi,0} := 1$, $r_{\phi,1} := g_\phi$ and for $k > 1$,

$$r_{\phi,k} := g_\phi^{q^{k-1}} r_{\phi,k-1} - (x^{q^{k-1}} - x)\Delta_\phi^{q^{k-2}} r_{\phi,k-2} \in \mathbb{F}_q[x]. \tag{2}$$

Gekeler [18, Eq 3.6, Prop 3.7] showed that $r_{\phi,k}$ is the value of the normalized Eisenstein series of weight $q^k - 1$ on ϕ and established Deligne's congruence for Drinfeld modules, which ascertains that for any irreducible f of degree $k \geq 1$ with $\Delta_\phi \neq 0 \mod f$, we have

$$h_{\phi,f} = r_{\phi,k} \mod f. \tag{3}$$

Hence $r_{\phi,k}$ is in a sense a lift to $\mathbb{F}_q[x]$ of all the Hasse invariants of ϕ at primes of degree k. Using the sequence $r_{\phi,k}$ allows us to define a polynomial $\bar{h}_{\phi,f}$ for arbitrary non-irreducible polynomials f, by setting

$$\bar{h}_{\phi,f} := \gcd\left(r_{\phi,\deg(f)} \bmod f, r_{\phi,\deg(f)+1} \bmod f\right). \tag{4}$$

Our definition will be justified in light of Lemma 3.1; note that unlike in the irreducible case, we define $\bar{h}_{\phi,f}$ only up to a non zero $\mathbb{F}_q$ multiple, which will suffice for our purpose. The following subsections give our algorithm to compute $\bar{h}_{\phi,f}$ for an arbitrary squarefree f, thereby proving Theorem 1.1.

2.2 Some key subroutines

We summarize here the main results we will need; most of them are well-known, and originate from work of von zur Gathen-Shoup [16] or Kaltofen-Shoup [23]. A key ingredient is the use of *modular composition*, that is, the operation $(f, g, h) \mapsto f(g) \bmod h$, since several operations related to the Frobenius map can be computed efficiently using this as a subroutine.

For f, g, h of degree n, the best known algorithm for modular composition was for long Brent and Kung's result [4], with a cost of $O(n^{(\omega+1)/2})$ base field operations; improvements using fast rectangular matrix multiplication followed in the work of Huang and Pan [22]. More recently, Kedlaya and Umans showed that in a *boolean* model, there exist algorithms of cost close to *linear*: for any $\varepsilon > 0$, there is an algorithm for modular composition of degree n polynomials over $\mathbb{F}_q$ that takes $n^{1+\varepsilon}(\log q)^{1+o(1)}$ bit operations.

As of now, algorithms based on Brent and Kung's result have been implemented on a variety of platforms [33, 2, 20], and still outperform implementations of the Kedlaya-Umans algorithm. As a result, we decided to give two variants of our main algorithm, using these two possible key subroutines.

For the rest of this section, consider a squarefree polynomial f in $\mathbb{F}_q[x]$, define $\mathbb{K} = \mathbb{F}_q[x]/f$ and let ξ be the image of x in $\mathbb{K}$. Let $\tau : \mathbb{K} \to \mathbb{K}$ be the $\mathbb{F}_q$-linear qth-power Frobenius map $a \mapsto a^q$; since f is squarefree, τ is invertible. It will then be convenient to define $\xi_i := \tau^i(\xi)$, for i in $\mathbb{Z}$; then, for any a in $\mathbb{K}$ and i in $\mathbb{Z}$, $a(\xi_i)$ is well-defined (as $A(\xi_i) \in \mathbb{K}$, for an arbitrary lift A of a to $\mathbb{F}_q[x]$) and satisfies $a(\xi_i) = \tau^i(a)$. In particular, for any i, j in $\mathbb{Z}$, the relation $\xi_i(\xi_j) = \xi_{i+j}$ holds.

The following items describe subroutines needed in our main algorithm. With the partial exception of the last one (see the remark below), all of them are known results.

1. Computing $\xi_1 = \xi^q$ is done by repeated squaring, using $O(\mathsf{M}(n) \log q)$ operations in $\mathbb{F}_q$.

2. Once $\xi_1 = \xi^q$ is known, any ξ_r, $r \geq 0$, can be computed at the cost of $O(\log r)$ modular compositions, using the relation $\xi_i(\xi_j) = \xi_{i+j}$ stated above.

3. Given ξ_i ($i \geq 0$), we can compute ξ_{-i} by solving the linear equation $\xi_{-i}(\xi_i) = \xi$. This can be done using *transposed* modular composition, with a cost identical (up to a constant factor) to that of modular composition itself [34, 12].

4. Given $(a_0, \ldots, a_{k-1})$ in $\mathbb{K}$ and ξ_i, for some $i \in \mathbb{Z}$, consider the question of computing $(\tau^i(a_0), \ldots, \tau^i(a_{k-1})) = (a_0(\xi_i), \ldots, a_{k-1}(\xi_i))$. In the boolean model, the cost is

$kn^{1+\varepsilon}(\log q)^{1+o(1)}$ bit operations, which is essentially optimal. In the algebraic model, for $k = O(n)$, Lemma 3 in [23] shows how to do this in $O(k^{(\omega-1)/2}n^{(\omega+1)/2})$ operations in $\mathbb{F}_q$ instead of $O(kn^{(\omega+1)/2})$, by exploiting the fact that the second argument is the same for all instances.

5. Given integers $u, \ell, m \in O(n)$, and ξ_1, one can compute $(\xi_{-u}, \xi_{-(u+\ell)}, \dots, \xi_{-(u+(m-1)\ell)})$ using $O((m^{(\omega-1)/2} + \log n)n^{(\omega+1)/2})$ operations in $\mathbb{F}_q$, resp. $mn^{1+\varepsilon}(\log q)^{1+o(1)}$ bit operations. The latter estimate is straightforward; for the former, we follow [23, Lemma 4], which shows (in our notation) how to compute $\xi_\ell, \xi_{2\ell}, \dots, \xi_{(m-1)\ell}$. We work here with negative indices, but this hardly changes the procedure.

We first compute $\xi_{-\ell}$, at the cost of $O(\log \ell)$ modular compositions. Then, for $k \geq 1$, assuming that we know $\xi_{-\ell}, \xi_{-2\ell}, \dots, \xi_{-k\ell}$, we deduce $\xi_{-(k+1)\ell}, \xi_{-(k+2)\ell}, \dots, \xi_{-2k\ell}$ from the relation $\xi_{-(k+j)\ell} = \xi_{-j\ell}(\xi_{-k\ell})$, by means of item **4**. We repeat this process for $k = 1, 2, 4, \dots$, stopping at the first power of two greater than or equal to m. At this stage, we know $\xi_{-\ell}, \dots, \xi_{-(m-1)\ell}$. To conclude, we compute ξ_{-u} at the cost of $O(\log u)$ modular compositions, and finally $\xi_{-\ell}(\xi_{-u}), \dots, \xi_{-(m-1)\ell}(\xi_{-u})$. The total cost is $O(m^{(\omega-1)/2}n^{(\omega+1)/2})$ operations in $\mathbb{F}_q$.

6. For u, ℓ, m as above, given $(a_0, \dots, a_{m-1})$ and ξ_1, we finally show how to compute the product $\tau^{u+(m-1)\ell}(a_{m-1}) \cdots \tau^u(a_0)$. Here, we actually take all a_i as 2×2 matrices over $\mathbb{K}$, since this is what we will need below (this has no impact on the algorithm description, except that we must account for the non-commutativity of the product). As above, in the boolean model, the cost is easily seen to be $mn^{1+\varepsilon}(\log q)^{1+o(1)}$ bit operations.

To discuss the algorithm in the algebraic model, without loss of generality, we assume that m is a power of 2, say $m = 2^t$. First, we replace our input by $(a_0(\xi_u), \dots, a_{m-1}(\xi_u))$. Let us then set $\mu = m$ and $\zeta = \xi_\ell$. For $k = 0, \dots, t-1$, we do the following: for $i = 0, \dots, \mu/2 - 1$, replace a_i by $a_{2i+1}(\zeta)a_{2i}$; then let $\zeta = \zeta(\zeta)$ and $\mu = \mu/2$. At the end of the loop, the first entry in the sequence is the requested output.

Computing ξ_ℓ and ξ_u takes $O(\log n)$ modular compositions. The initial composition by ξ_u takes time $O(m^{(\omega-1)/2}n^{(\omega+1)/2})$, by item **4**; for an index k in the main loop, the cost is similarly $O((m/2^k)^{(\omega-1)/2}n^{(\omega+1)/2})$; The overall runtime is thus $O((m^{(\omega-1)/2} + \log n)n^{(\omega+1)/2})$ operations in $\mathbb{F}_q$.

Remark. The description we give for item **6** answers a question in [23, Section 3.2]: the authors proved the existence of an algorithm to compute (in our notation) an expression of the form $a_0 + \tau^\ell(a_1) + \cdots + \tau^{(m-1)\ell}(a_{m-1})$ in $O(m^{(\omega-1)/2}n^{(\omega+1)/2})$ base field operations, but left the actual description of such an algorithm as an open question (in that context, $\log n$ is negligible in front of $m^{(\omega-1)/2}$, and the a_i's are actually in $\mathbb{K}$). The procedure we gave above can be adapted to the computation of such a sum, simply by replacing all products by additions in the main loop.

2.3 Efficient computation of $\bar{h}_{\phi,f}$

Given a squarefree f in $\mathbb{F}_q[x]$ as above, with $\deg(f) = n$, and an elliptic module ϕ over $\mathbb{F}_q(x)$, we present an efficient algorithm to compute $r_{\phi,n} \bmod f$ and $r_{\phi,n+1} \bmod f$, thereby yielding $\bar{h}_{\phi,f} = \gcd(r_{\phi,n} \bmod f, r_{\phi,n+1} \bmod f)$. The latter gcd can be computed in quasi-linear time, namely $O(\mathsf{M}(n)\log(n))$ operations in $\mathbb{F}_q$, so we will not discuss it further.

Our strategy to compute $r_{\phi,n} \bmod f$ and $r_{\phi,n+1} \bmod f$ is to first phrase the recurrence in matrix form with entries being polynomials modulo f. We then observe that solving the recurrence amounts to computing the product of a carefully constructed sequence of matrices twisted by the Frobenius action. The final step is to construct a polynomial with matrix coefficients, whose evaluations allow us to rapidly compute the aforementioned product; the evaluations are then computed using a fast multipoint evaluation algorithm.

The overall structure of the algorithm is similar to the distinct degree factorization algorithm of [23], with the polynomial matrix described above being akin to the "interval polynomial" used in that algorithm. Another inspiration for our algorithm is computation with simpler recurrences, that go back to Strassen's deterministic integer factorization algorithm [35] and Chudnovsky and Chudnovsky's algorithm for linear recurrences with polynomial coefficients [8]. These baby-steps / giant steps techniques allow one to compute the nth term in a sequence defined by such a recurrence, for instance $u_0 = 1, u_{n+1} = (n+1)u_n$, in $O\tilde{}(\sqrt{n})$ base field operations. These ideas were subsequently refined and applied to the computation of the Hasse-Witt matrix of hyperelliptic curves in [3].

With $\mathbb{K} = \mathbb{F}_q[x]/f$ and ξ as before, we have to compute $\rho_n := r_{\phi,n}(\xi)$ and $\rho_{n+1} := r_{\phi,n+1}(\xi)$, where the sequence $r_{\phi,k}$ is from (2); equivalently,

$$\rho_0 = 1, \quad \rho_1 = \gamma, \quad \rho_k = \gamma^{q^{k-1}} \rho_{k-1} - (\xi^{q^{k-1}} - \xi)\delta^{q^{k-2}} \rho_{k-2}, \tag{5}$$

with $\gamma = g_\phi(\xi) \in \mathbb{K}$ and $\delta = \Delta_\phi(\xi) \in \mathbb{K}$. Computing all terms $\rho_0, \rho_1, \dots, \rho_{n+1}$ takes $\Omega(n^2)$ operations in $\mathbb{F}_q$, since merely writing down each ρ_i involves $\Theta(n)$ operations. The algorithm in this section takes subquadratic time.

In the case of elliptic curves, the Hasse invariant is obtained as the element of index $(p-1)/2$ in an order-2 recurrence with polynomial coefficients. Seeing the similarity with Eq. (5), it is natural to adapt these ideas to our context. This is however not entirely straightforward. Indeed, given ρ_{k-1} and ρ_{k-2}, computing ρ_k now boils down to applying powers of the Frobenius endomorphism, together with a few polynomial multiplications / additions.

The recurrence (5) can be written as

$$\begin{bmatrix} \rho_{k-1} \\ \rho_k \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ (\xi - \xi^{q^{k-1}})\delta^{q^{k-2}} & \gamma^{q^{k-1}} \end{bmatrix} \begin{bmatrix} \rho_{k-2} \\ \rho_{k-1} \end{bmatrix}.$$

Let as before $\tau : \mathbb{K} \to \mathbb{K}$ be the $\mathbb{F}_q$-linear qth-power Frobenius map, and define the following sequence of matrices in $\mathscr{M}_2(\mathbb{K})$:

$$A_k := \begin{bmatrix} 0 & 1 \\ (\xi - \xi^{q^{k+1}})\delta^{q^k} & \gamma^{q^{k+1}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ (\xi - \tau^{k+1}(\xi))\tau^k(\delta) & \tau^{k+1}(\gamma) \end{bmatrix};$$

then, we have

$$\begin{bmatrix} \rho_{k-1} \\ \rho_k \end{bmatrix} = A_{k-2} A_{k-3} \cdots A_0 \begin{bmatrix} 1 \\ \gamma \end{bmatrix}.$$

Given integers $m \le m'$, we show how to compute the product

$$B_{m,m'} := A_{m'-1} \cdots A_m \ \in \ \mathscr{M}_2(\mathbb{K}),$$

for then we can read off ρ_n and ρ_{n+1} from $B_{0,n} \left[\begin{smallmatrix} 1 \\ \gamma \end{smallmatrix}\right]$. We need the extra flexibility of starting the product at index m in the algorithm; concretely, we will rely on the relation $B_{m',m''} B_{m,m'} = B_{m,m''}$, for any integers $m \le m' \le m''$.

Extend the mapping τ to the (non-commutative) polynomial ring $\mathscr{M}_2(\mathbb{K})[Y]$ by leaving Y fixed and acting on the coefficient matrices entry-wise. Let further

$$\mathcal{A} := \begin{bmatrix} 0 & 1 \\ -\tau(\xi)\delta & \tau(\gamma) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \delta & 0 \end{bmatrix} Y \ \in \ \mathscr{M}_2(\mathbb{K})[Y],$$

and for $\mathcal{M} \in \mathscr{M}_2(\mathbb{K})[Y]$ and $\zeta \in \mathbb{K}$, let $\mathcal{M}(\zeta)$ denote the image of $\mathcal{M}$ under the substitution

$$Y \longmapsto D_\zeta = \begin{bmatrix} \zeta & 0 \\ 0 & \zeta \end{bmatrix};$$

in particular, $\mathcal{M} \mapsto \mathcal{M}(\zeta)$ is a ring homomorphism, since D_ζ is in the center of $\mathscr{M}_2(\mathbb{K})$. Then, for any $k \geq 0$, we have

$$A_k = \tau^k(\mathcal{A})(\xi).$$

Choose an integer $\ell \leq n+1$ and define

$$\mathcal{B} := \tau^{\ell-1}(\mathcal{A}) \cdots \tau(\mathcal{A})\mathcal{A} \ \in \ \mathscr{M}_2(\mathbb{K})[Y],$$

so that in particular $\mathcal{B}(\xi) = A_{\ell-1} \cdots A_1 A_0$. More generally, we use the fact that for all integers i, j, we have

$$A_{i+j} = \tau^{i+j}(\mathcal{A})(\xi) = \tau^i\big(\tau^j(\mathcal{A})(\xi_{-i})\big)$$

to deduce that for all $i \geq 1$,

$$\tau^i(\mathcal{B}(\xi_{-i})) = A_{i+\ell-1} \cdots A_{i+1} A_i = B_{i,i+\ell}.$$

In particular, for any integer m, $B_{0,m\ell}$ can be computed as the product of the matrices

$$\tau^{(m-1)\ell}\big(\mathcal{B}(\xi_{-(m-1)\ell})\big), \ldots, \tau^{\ell}\big(\mathcal{B}(\xi_{-\ell})\big), \mathcal{B}(\xi);$$

more generally, for any integers m, u, we have

$$B_{u,u+m\ell} = \tau^{u+(m-1)\ell}\big(\mathcal{B}(\xi_{-(u+(m-1)\ell)})\big) \cdots \tau^{u+\ell}\big(\mathcal{B}(\xi_{-(u+\ell)})\big)\tau^u\big(\mathcal{B}(\xi_{-u})\big).$$

Suppose that we know $B_{0,u}$, for some $u < n$. As in [23], let β be an arbitrary constant in $(0,1)$, and define $\ell = \lceil (n-u)^\beta \rceil$, $m = \lfloor (n-u)/\ell \rfloor$. Then, $v = u + m\ell$ satisfies $v \leq n$ and $n - v \leq (n-u)^\beta$. The discussion above suggests Algorithm 1 for computing $B_{u,v}$, from which we can deduce $B_{0,v} = B_{u,v} B_{0,u}$.

Algorithm 1 Main subroutine for the Hasse invariant

Input: f squarefree in $\mathbb{F}_q[x]$, $\delta, \gamma \in \mathbb{K} = \mathbb{F}_q[x]/f$, integers u, n, with $u < n$
Output: $B_{u,v}$ as defined above

1. Let $\ell := \lceil (n-u)^\beta \rceil$ and $m := \lfloor (n-u)/\ell \rfloor$
2. Compute $\mathcal{B} = \tau^{\ell-1}(\mathcal{A}) \cdots \tau(\mathcal{A})\mathcal{A}$
3. Compute $\xi_{-(u+i\ell)} = \tau^{-(u+i\ell)}(\xi)$ for $0 \leq i < m$
4. Compute $\beta_i = \mathcal{B}(\xi_{-(u+i\ell)})$ for $0 \leq i < m$
5. **return** the product $\tau^{u+(m-1)\ell}(\beta_{m-1}) \cdots \tau^u(\beta_0)$

Correctness of the algorithm follows from the preceding remarks. We will give two different runtime analyses, in respectively the algebraic and boolean model, where the main difference lies in the cost of modular composition.

- In the whole algorithm, we compute once $\xi_1 = \tau(\xi)$; the cost is $O(\mathsf{M}(n)\log(q))$ operations in $\mathbb{F}_q$.

- A first solution for Step 2 is to compute $\tau(\mathcal{A}), \ldots, \tau^{\ell-1}(\mathcal{A})$ (using $\ell - 1$ successive applications of the Frobenius) and multiply the results. The successive applications of τ are done using repeated squaring, and the subsequent multiplications using a 2×2 matrix version of the subproduct tree [15, Chapter 10]. Multiplication of 2×2 polynomial matrices in degree k over $\mathbb{K}$ takes $O(\mathsf{M}(kn))$ operations in $\mathbb{K}$ (using Kronecker substitution), so the cost is $O(\ell\mathsf{M}(n)\log q + \mathsf{M}(\ell n)\log \ell)$ operations in $\mathbb{F}_q$.

  An alternative is to perform Step 2 recursively: given $\tau^{i-1}(\mathcal{A})\cdots\mathcal{A}$, it takes one application of τ^i (resp. of τ) and one matrix multiplication in $\mathscr{M}_2(\mathbb{K})[Y]$ to compute $\tau^{2i-1}(\mathcal{A})\cdots\mathcal{A}$, resp. $\tau^i(\mathcal{A})\cdots\mathcal{A}$. The power-of-$\tau$ map is computed using the iterated Frobenius algorithm of von zur Gathen and Shoup [16]. In the algebraic model, the cost of modular composition makes this solution inferior to the one in the previous paragraph. In the boolean model, since ξ^q is known, the cost becomes $\ell n^{1+\varepsilon}(\log q)^{1+o(1)}$ bit operations.

- Items **5** and **6** in the previous subsection show that Step 3 and 5 can be done using $O(m^{(\omega-1)/2}n^{(\omega+1)/2})$ operations in $\mathbb{F}_q$, resp. $mn^{1+\varepsilon}(\log q)^{1+o(1)}$ bit operations.

- Step 4 can be done using multipoint evaluation [15]. We are evaluating a 2×2 polynomial matrix of degree at most ℓ at m points. We first build the subproduct tree at the given points, then reduce each entry of the matrix modulo the root polynomial of the tree, and apply the "going down the tree" procedure of [15, Chapter 10]. Altogether, this takes $O(\mathsf{M}(mn)\log m + \mathsf{M}(\ell n))$ operations in $\mathbb{F}_q$.

Altogether, in the algebraic model, we obtain the following result.

Proposition 2.1. *Algorithm 1 can be implemented so as to run in*

$$O((n-u)^{(1-\beta)(\omega-1)/2}n^{(\omega+1)/2} + \mathsf{M}((n-u)^{\beta}n)\log qn)$$

operations in $\mathbb{F}_q$, where ω is the matrix multiplication exponent.

In the boolean model, we set $\beta = 1/2$. Then, each of the steps 2, 3, 4 and 5 can each be performed in $n^{3/2+\varepsilon}(\log q)^{1+o(1)}$ bit operations, whereas the initial computation of ξ^q takes $n^{1+\varepsilon}(\log q)^{2+o(1)}$ bit operations. To summarize, we have the following result.

Proposition 2.2. *Algorithm 1 can be implemented so as to run in*

$$O((n-u)^{1/2}n^{1+\varepsilon}(\log q)^{1+o(1)} + n^{1+\varepsilon}(\log q)^{2+o(1)})$$

bit operations.

To conclude the proof of Theorem 1.1, we apply the previous results with input $u = 0$ and n, and obtain $B_{0,v}$, for v as defined previously. If $v = n$, we are done, otherwise we re-enter the algorithm with input v and n (so that $n - v \le n^{\beta}$), and so on. Starting from Proposition 2.1, we see that the cost of the first call dominates the overall runtime, so the first item in Theorem 1.1 is proved. Using Proposition 2.2, the situation is similar, up to the total contribution of the second term in the sum, since a factor $\log\log n$ appears; however, we can absorb it in the $1 + \varepsilon$ exponent and the second claim in Theorem 1.1 follows.

3 A polynomial factoring algorithm

We can now give our elliptic module algorithm for polynomial factorization. As input, we are given $f \in \mathbb{F}_q[x]$ of degree n; without loss of generality, we may assume that f is squarefree [25, 37], that is, does not contain a square of an irreducible polynomial as a factor.

Let $\phi : \mathbb{F}_q[x] \to \mathbb{F}_q(x)\{\tau\}$ be an elliptic module over $\mathbb{F}_q(x)$ and suppose that $\gcd(f, \Delta_\phi) = 1$. The following lemma is the key of our algorithm.

Lemma 3.1. *Let g be an irreducible factor of f. Then ϕ has supersingular reduction at g if and only if $\bar{h}_{\phi,f} \bmod g = 0$.*

Proof. First, note that by assumption, ϕ has good reduction at g. Let $k := \deg(g)$ and $n := \deg(f)$ and for $j \geq 0$ let $r_{\phi,j} \in \mathbb{F}_q[x]$ be the sequence as in recurrence (2), defined with respect to ϕ.

Assume ϕ has supersingular reduction at g, that is, $h_{\phi,g} = 0$ (recall that this is a polynomial defined modulo g). By Deligne's congruence, $r_{\phi,k} \bmod g = 0$. Since g divides $x^{q^k} - x$, the recurrence relation (2) implies $r_{\phi,k+1} \bmod g = 0$. Since this recurrence has order 2, this further yields

$$r_{\phi,j} \bmod g = 0, \quad j \geq k. \tag{6}$$

In particular, $r_{\phi,n} \bmod g = 0$ and $r_{\phi,n+1} \bmod g = 0$; this implies $\bar{h}_{\phi,f} \bmod g = 0$, since g divides f.

Conversely, assume $\bar{h}_{\phi,f} \bmod g = 0$. To prove that ϕ has supersingular reduction at g, using Deligne's congruence, it suffices to show $r_{\phi,k} \bmod g = 0$. Since $\bar{h}_{\phi,f} \bmod g = 0$ and g divides f, we have $r_{\phi,n} \bmod g = r_{\phi,n+1} \bmod g = 0$. In particular, if $g = f$, we are done, since then $k = n$.

Else, let m be the smallest positive integer greater than k such that $r_{\phi,m} \bmod g = 0$ and $r_{\phi,m+1} \bmod g = 0$; such an m exists, since then $k < n$. From the recurrence (2),

$$r_{\phi,m+1} \mod g = g_\phi^{q^m} r_{\phi,m} - \left(x^{q^m} - x\right) \Delta_\phi^{q^{m-1}} r_{\phi,m-1} \mod g;$$

hence,

$$\left(x^{q^m} - x\right) \Delta_\phi^{q^{m-1}} r_{\phi,m-1} \mod g = 0.$$

Since ϕ has good reduction at ϕ, we may conclude that either $r_{\phi,m-1} \mod g = 0$ or k divides m. If $r_{\phi,m-1} = 0 \bmod g$, then by the minimality of m, we may conclude $m - 1 = k$, that is, $r_{\phi,k} \bmod g = 0$, proving our claim. If k divides m, then $m - k \geq k$ (since $m > k$). By [11, Lemma 2.3],

$$r_{\phi,m} \bmod g = r_{\phi,k} r_{\phi,m-k}^{q^k} \bmod g, \quad r_{\phi,m+1} \bmod g = r_{\phi,k} r_{\phi,m+1-k}^{q^k} \bmod g,$$

implying either $r_{\phi,k} \mod g = 0$ (proving our claim) or $r_{\phi,m+1-k} \mod g = r_{\phi,m-k} \mod g = 0$. Since $m - k \geq k$, for the latter case to not contradict the minimality of m, $m - k$ has to equal k, implying $r_{\phi,k} \mod g = 0$, thereby proving our claim. □

This suggests that we could use an elliptic module ϕ in a polynomial factorization algorithm to separate supersingular primes from those that are not. For most elliptic modules, the density of supersingular primes is too small for this to work. However, for a special class, elliptic modules with complex multiplication, the density of supersingular primes is 1/2.

Let $L = \mathbb{F}_q(x)(\sqrt{d})$ be a quadratic extension of $\mathbb{F}_q(x)$, for some polynomial d in $\mathbb{F}_q[x]$. We say that L is *imaginary* if the prime $(1/x) \in \mathbb{F}_q(x)$ at infinity does not split in L; this is in particular the case if d is squarefree of odd degree (in which case the prime at infinity ramifies [30, Proposition 14.6]).

An elliptic module ϕ over $\mathbb{F}_q(x)$ is said to have *complex multiplication* by an imaginary quadratic extension $L/\mathbb{F}_q(x)$ if $\mathrm{End}_{\mathbb{F}_q(x)}(\phi) \otimes_{\mathbb{F}_q[x]} \mathbb{F}_q(x)$ is isomorphic to L, where $\mathrm{End}_{\mathbb{F}_q(x)}(\phi)$ is the *ring of* *endomorphisms* of ϕ, that is, the ring of all elements of $\mathbb{F}_q(x)\{\tau\}$ that commute with ϕ_x. For ϕ with complex multiplication by $L/\mathbb{F}_q(x)$ and an irreducible polynomial $f \in \mathbb{F}_q[x]$ such that $\langle f \rangle$ is unramified in $L/\mathbb{F}_q(x)$, f is supersingular if and only if it $\langle f \rangle$ is inert in $L/\mathbb{F}_q(x)$.

This suggests the following strategy to factor a monic squarefree polynomial $f \in \mathbb{F}_q[x]$. Say f factors into monic irreducibles as $f = \prod_i f_i$. Pick an elliptic module ϕ with complex multiplication by some imaginary quadratic extension $L/\mathbb{F}_q(x)$ and compute $\bar{h}_{\phi,f}$. By Lemma 3.1, we get that

$$\gcd(\bar{h}_{\phi,f}, f) = \prod_{\langle f_i \rangle \text{ inert in } L/\mathbb{F}_q(x)} f_i \tag{7}$$

is a factor of f. Since for every degree, roughly half the primes of that degree are inert in $L/\mathbb{F}_q(x)$, the factorization thus obtained is likely to be non trivial. Repeating the process for the resulting factors leads to a complete factorization of f.

It remains to construct elliptic modules with complex multiplication. Our strategy is to pick an $a \in \mathbb{F}_q$ at random and construct an elliptic module ϕ with complex multiplication by the imaginary quadratic extension $\mathbb{F}_q(x)(\sqrt{d})$ of discriminant $d := x - a$. From [13] (see also [32, Theorem 6]), the elliptic module ϕ' with

$$g_{\phi'}(x) := \sqrt{d} + \sqrt{d}^q, \quad \Delta_{\phi'} := 1$$

has complex multiplication by $\mathbb{F}_q(x)(\sqrt{d})$. However, ϕ' has the disadvantage of not being defined over $\mathbb{F}_q[x]$, since $g_{\phi'}$ is not in $\mathbb{F}_q[x]$. We construct an alternate ϕ that is isomorphic to ϕ' but defined over $\mathbb{F}_q[x]$. There is a notion of J-invariant for elliptic modules (see e.g. [18]); the J-invariant of ϕ' is

$$J_{\phi'} := \frac{g_{\phi'}^{q+1}}{\Delta_{\phi'}} = d^{\frac{q+1}{2}} \left(1 + d^{\frac{q-1}{2}}\right)^{q+1}.$$

Now let ϕ be the elliptic module defined by

$$g_\phi = J_{\phi'}, \quad \Delta_\phi = J_{\phi'}^q$$

The J-invariants of ϕ and ϕ' are the same, which implies that ϕ' and ϕ are isomorphic. Further, ϕ is defined over $\mathbb{F}_q[x]$. In summary, ϕ has complex multiplication by $\mathbb{F}_q(x)(\sqrt{d})$ and is defined over $\mathbb{F}_q[x]$.

Remark. When q is even, the construction of Schweizer [32, Theorem 6] gives the required elliptic modules with complex multiplication. Pick an $(a, b) \in \mathbb{F}_q^\times \times \mathbb{F}_q$ at random and set $d = ax + b$. The elliptic module ϕ with J-invariant

$$J_\phi = \left(1 + d + \ldots + d^{2^{m-2}} + d^{2^{m-1}}\right)^{q+1} / a$$

has complex multiplication by the ring $\mathbb{F}_q[x][w]$ in the Artin-Schreier extension $\mathbb{F}_q(x)(w)$, where $w^2 + w = d$. We can take ϕ to be the elliptic module defined by

$$g_\phi = J_\phi, \quad \Delta_\phi = J_\phi^q,$$

ensuring ϕ is defined over $\mathbb{F}_q[x]$. For a further discussion of characteristic 2, see Remark 3 below.

We now state our randomized algorithm to factor polynomials over finite fields using elliptic modules with complex multiplication.

Algorithm 2 Polynomial factorization

Input: Monic squarefree $f \in \mathbb{F}_q[x]$ of degree n

Output: The irreducible factors of f

1. If f is irreducible then output f and return
2. Remove the linear factors of f and output them
3. Pick $a \in \mathbb{F}_q$ uniformly at random and compute
   $d := x - a$,
   $J := d^{\frac{q+1}{2}} \left(1 + d^{\frac{q-1}{2}}\right)^{q+1} \bmod f$,
   $g_\phi := J \bmod f$ and $\Delta_\phi := J^q \bmod f$
4. Compute $\gamma := \gcd(\bar{h}_{\phi,f}, f)$ and recursively factor γ and f/γ.

The irreducibility test in Step 1 can be performed in $O(n^{(\omega+1)/2}(\log n)^2 + \mathsf{M}(n)\log q)$ operations in $\mathbb{F}_q$ [15], or $n^{1+\varepsilon}(\log q)^{2+o(1)}$ bit operations. In Step 2, all the linear factors of f are found and removed using a root finding algorithm; it takes an expected $O(\mathsf{M}(n)\log n \log(nq))$ operations in $\mathbb{F}_q$ [15], or $n^{1+\varepsilon}(\log q)^{2+o(1)}$ bit operations (this step needs only be done once in the whole algorithm).

In Step 3, we choose $a \in \mathbb{F}_q$ at random and construct a Drinfeld module ϕ with complex multiplication by $\mathbb{F}_q(x)(\sqrt{x-a})$. The primes that divide Δ_ϕ are precisely $\{(x-b), b \in \mathbb{F}_q, \sqrt{b-a} \notin \mathbb{F}_q\} \cup \{x-a\}$. Hence, we might have run into issues of bad reduction if f had linear factors; it is to prevent this that we performed root finding in Step 2. The computation of g_ϕ and Δ_ϕ takes $O(\mathsf{M}(n)\log q)$ operations in $\mathbb{F}_q$.

As mentioned in Eq. (7), $\gcd(\bar{h}_{\phi,f}, f)$ is the product of all irreducible factors of degree greater than 1 of f that are supersingular with respect to ϕ. Thus, our algorithm separates the irreducible factors supported at the supersingular primes from those supported at the ordinary primes. In the following, we show that for an elliptic module chosen randomly as in Step 3, the splitting of f in Step 4 is random enough to ensure that the recursion depth is $O(\log n)$.

We assume as in the statement of Theorem 1.2 that $\bar{h}_{\phi,f}$ is computed using $H(n,q)$ operations in $\mathbb{F}_q$, resp. $H^*(n,q)$ bit operations. Since the expected depth of the recursion is logarithmic, using the superlinearity assumptions on H, resp. H^* made in that theorem, the runtime of the algorithm is then an expected

- $O\tilde{\ }(H(n,q) + n^{(\omega+1)/2} + \mathsf{M}(n)\log q)$ operations in $\mathbb{F}_q$, or
- $O\tilde{\ }(H^*(n,q)) + n^{1+\varepsilon}(\log q)^{2+o(1)}$ bit operations;

this proves Theorem 1.2.

In the following lemma, we establish the claim above on the probability of finding a nontrivial factor of f. For the lemma to apply to the algorithm, we need to assume $\sqrt{q} \geq 12(n+2)$. This assumption can be made without loss of generality: if $\sqrt{q} < 12(n+2)$, we might choose to factor over a slightly larger field $\mathbb{F}_{q'}$ where q' is the smallest power of q such that $\sqrt{q'} > 12(n+2)$ and still recover the factorization over $\mathbb{F}_q$ (c.f. [28, Remark 3.2]). Further, the running times are only affected by logarithmic factors in n.

Lemma 3.2. *Suppose that f is not irreducible, without linear factors, and that $12(n+2) \leq \sqrt{q}$. Let ϕ be an elliptic module with complex multiplication by the imaginary quadratic extension $\mathbb{F}_q(x)(\sqrt{x-a})$, where $a \in \mathbb{F}_q$ is chosen at random. Then, with probability at least $1/4$, the factor h computed at Step 4 of the algorithm is non-trivial.*

Proof. Suppose that f admits two distinct monic irreducible factors f_1 and f_2, of respective degrees $k_1, k_2 > 1$. We prove that with probability at least $1/4$, exactly one of f_1 or f_2 is supersingular with respect to a randomly chosen ϕ, as constructed above.

Since k_1, k_2 are greater than 1, none of $\langle f_1\rangle, \langle f_2\rangle$ ramify in $\mathbb{F}_q(x)(\sqrt{x-a})$. Therefore, the probability that exactly one of $\langle f_1\rangle, \langle f_2\rangle$ is supersingular with respect to ϕ is the same as the probability that exactly one of them splits in $\mathbb{F}_q(x)(\sqrt{x-a})/\mathbb{F}_q(x)$.

For $i = 1, 2$, let $K_i := \mathbb{F}_q(x)(\alpha_i)$ be the hyperelliptic extension of $\mathbb{F}_q(x)$ obtained by adjoining a root α_i of $y^2 - f_i$. Depending on the values of q and k_1, k_2, by quadratic reciprocity over function fields [7], one of the following alternative holds:

(a) exactly one of $\langle f_1\rangle, \langle f_2\rangle$ splits in $\mathbb{F}_q(x)(\sqrt{x-a})$ if and only if $x - a$ splits in exactly one of K_1, K_2;

(b) exactly one of $\langle f_1\rangle, \langle f_2\rangle$ splits in $\mathbb{F}_q(x)(\sqrt{x-a})$ if and only if $x - a$ either splits in K_1 and K_2, or splits in none of them.

We are in case (a) if $q = 1 \bmod 4$ or $k_1 = k_2 \bmod 2$, and in case (b) for all other values of the parameters. Since f_1, f_2 are distinct, K_1 and K_2 are linearly disjoint over $\mathbb{F}_q(x)$. Further, K_1K_2 is Galois over $\mathbb{F}_q(x)$ with

$$\mathrm{Gal}(K_1K_2/\mathbb{F}_q(x)) \cong \mathrm{Gal}(K_1/\mathbb{F}_q(x)) \times \mathrm{Gal}(K_2/\mathbb{F}_q(x)) \cong \mathbb{Z}/2\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}.$$

For $\langle x - a\rangle$ to be neither totally split nor totally inert (case (a)), the Artin symbol

$$(\langle x - a\rangle, K_1K_2/\mathbb{F}_q(x)) \in \mathrm{Gal}(K_1K_2/\mathbb{F}_q(x))$$

has to be either $(0, 1)$ or $(1, 0)$ under the isomorphism $\mathrm{Gal}(K_1K_2/\mathbb{F}_q(x)) \cong \mathbb{Z}/2\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$; case (b) occurs when its value is either $(0, 0)$ or $(1, 1)$. Applying the effective Chebotarev density theorem over function fields [14, Proposition 6.4.8], for any $\eta = (\eta_1, \eta_2) \in \mathbb{Z}/2\mathbb{Z} \oplus \mathbb{Z}/2\mathbb{Z}$, we obtain that the number N_η of degree 1 primes in $\mathbb{F}_q(x)$ that are unramified in K_1K_2 and of symbol η satisfies

$$\left|N_\eta - \frac{q}{4}\right| \le \frac{1}{2}\left((4 + g(K_1K_2))q^{1/2} + 4q^{1/4} + (4 + g(K_1K_2))\right)$$

where $g(K_1K_2)$ is the genus of K_1K_2. Taking into account the prime at infinity, we obtain that in case (a), the number $N_{(\mathrm{a})}$ of degree one primes $\{\langle x - a\rangle, a \in \mathbb{F}_q\}$ that are neither totally inert nor totally split in K_1K_2, and that in case (b), the number $N_{(\mathrm{b})}$ of degree one primes $\{\langle x - a\rangle, a \in \mathbb{F}_q\}$ that are either totally inert or totally split in K_1K_2 satisfy

$$N_{(\mathrm{a})} \ge N_{(0,1)} + N_{(1,0)} - 1, \quad N_{(\mathrm{b})} \ge N_{(0,0)} + N_{(1,1)} - 1.$$

The previous inequalities imply

$$\begin{aligned} N_{(\mathrm{a})}, N_{(\mathrm{b})} &\ge \frac{1}{2}q - \left((4 + g(K_1K_2))q^{1/2} + 4q^{1/4} + (5 + g(K_1K_2))\right) \\ &\ge \frac{1}{2}q - 3(4 + g(K_1K_2))q^{1/2}. \end{aligned}$$

Using the Riemann-Hurwitz genus formula [30, Theorem 7.16], we now prove $g(K_1K_2) \le k_1 + k_2 - 2$. Indeed, after replacing $\mathbb{F}_q$ by a suitable algebraic extension $\mathbb{F}_{q'}$, the formula reads $2g(K_1K_2) - 2 = 4(0 - 2) + \sum_\alpha (e(\alpha) - 1)$, where the sum is over all points $\alpha \in \mathbb{P}^3(\mathbb{F}_{q'})$ lying on the projective closure C of the curve $V(y_1^2 - f_1(x), y_2^2 - f_2(x))$ at which the first-factor projection $C \to \mathbb{P}^1(\mathbb{F}_{q'})$ ramifies (here, $e(\alpha)$ is the ramification index). All roots of either f_1 or f_2 (which are all pairwise distinct) give two such points, each of them with ramification index 2; the contribution of the point at infinity in $\mathbb{P}^1(\mathbb{F}_{q'})$ is at most $4 - 1 = 3$, so that we have $2g(K_1K_2) - 2 \le -8 + 2(k_1 + k_2) + 3$, which implies our claim.

Using the inequality $k_1 + k_2 \leq n$, we deduce

$$N_{(\mathrm{a})}, N_{(\mathrm{b})} \geq \frac{1}{2}q - 3(n+2)q^{1/2},$$

so that in either case, the probability of finding a non-trivial factorization of f is at least $1/2 - 3(n+2)/q^{1/2}$. Since $12(n+2) \leq q^{1/2}$, this probability is at least $1/4$. □

Remark. There is an obstruction to an even characteristic analogue of Lemma 3.2 being true. Let q be even and suppose ϕ is an elliptic module with complex multiplication by the ring $\mathbb{F}_q[x][w]$ in an Artin-Schreier extension, where $(a, b) \in \mathbb{F}_q^\times \times \mathbb{F}_q$ is chosen at random and $w^2 + w = ax + b$. Let f have distinct monic irreducible factors f_1, f_2 of respective degrees $k_1 > 1, k_2 > 1$. Since k_1, k_2 are greater than 1, none of $\langle f_1 \rangle, \langle f_2 \rangle$ ramify in $\mathbb{F}_q(x)(w)$. Therefore, the probability that exactly one of $\langle f_1 \rangle, \langle f_2 \rangle$ is supersingular with respect to ϕ is the same as the probability that exactly one of them splits in $\mathbb{F}_q(x)(w)/\mathbb{F}_q(x)$. The latter is equivalent to exactly one of the two equations

$$w^2 + w = ax + b \bmod f_1(x), \;\; w^2 + w = ax + b \bmod f_2(x) \tag{8}$$

possessing a solution $w \in \mathbb{F}_q[x]$. Analogous to the proof of Lemma 3.2, one may look to reciprocity laws for Artin-Schreier extensions to phrase this condition in terms of polynomial congruences involving f_1 and f_2 modulo $ax + b$. However, the reciprocity law (due to Hasse [21], see also [9, Theorem 4.2]) dictates that the existence of solutions to the equations (8) depends not on congruences modulo $ax + b$ but only on the $(k_1 - 1)^{th}$ degree (resp. $(k_2 - 1)^{th}$) coefficient of f_1 (resp. f_2) and the parity of the degrees k_1 and k_2. In particular, by [9, Theorem 4.2] or [9, Example 4.1], if k_1 and k_2 have the same parity and the $(k_1 - 1)^{th}$ coefficient of f_1 equals the $(k_2 - 1)^{th}$ degree coefficient of f_2, then for no $(a, b) \in \mathbb{F}_q^\times \times \mathbb{F}_q$ does exactly one of the equations in (8) has a solution. Hence, our algorithm cannot distinguish the factors f_1 and f_2. It remains an open question as to if our algorithmic framework can be modified to handle this situation.

On the other hand, we may argue that this is the only obstruction. Say $f_1 = x^{k_1} + c_1 x^{k_1 - 1} + \cdots$ and $f_2 = x^{k_2} + c_2 x^{k_2 - 1} + \cdots$, where $c_1 \neq c_2$. Pick an elliptic module ϕ with complex multiplication by the ring $\mathbb{F}_q[x][w]$ in an Artin-Schreier extension, where $a \in \mathbb{F}_q^\times$ is chosen at random and $w^2 + w = ax$. By [9, Example 4.1], $w^2 + w = x \bmod f_1(x)$ has a solution $w \in \mathbb{F}_q[x]$ if and only if $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}(ac_1) = 0$, where $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}$ is the trace from $\mathbb{F}_q$ to $\mathbb{F}_2$. Likewise, $w^2 + w = x \bmod f_1(x)$ has a solution $w \in \mathbb{F}_q[x]$ if and only if $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}(ac_2) = 0$. For our algorithm to separate f_1 and f_2, it suffices for $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}(a(c_1 + c_2))$ to equal 1 with constant probability. Since $c_1 \neq c_2$, $c_1 + c_2 \neq 0$ and $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_2}(a(c_1 + c_2))$ does equal 1 with probability at least $1/2$.

4 Implementation and example

We implemented Algorithms 1 and 2 in C++ using NTL [33]; the implementation source code can be found at https://github.com/javad-doliskani/supersingular_drinfeld_factoring. After running the algorithm on 10^3 random input polynomials, we observed that the splitting $\gcd(\bar{h}_{\phi,f}, f)$ is almost always nontrivial when f is reducible; this confirms the behavior expected from theory.

We also observed that in practice one does not need to impose the condition $\sqrt{q} \geq 12(n+2)$ for splitting a polynomial of degree n over $\mathbb{F}_q$; this remedies the need for working in an auxiliary extension over $\mathbb{F}_q$. Recall that Lemma 3.2 uses that assumption to bound from below the probability of finding a non-trivial factorization; we expect that the recursion depth be logarithmic in n also for smaller values of q. At worst, one may first attempt splitting f over $\mathbb{F}_q$ and in case of repeated failures, we may then switch to factorization over an extension; this was not implemented.

In terms of runtime, our algorithm is slower that NTL's built-in factorization routine, by what is roughly a constant factor; this is not entirely surprising, since working with 2×2 matrices induces a non-negligible overhead for our algorithm. To show the behaviour of the algorithm in practice, we give in this section an example output of Algorithm 2 on input a small degree polynomial.

Example. Consider the randomly selected squarefree polynomial

$$f = 2 + 6x + 5x^3 + 4x^4 + 6x^5 + 2x^7 + 3x^8 + 3x^9 + x^{10}$$

in $\mathbb{F}_7[x]$. The algorithm starts by checking for linear factors; f has none. So, in the next step it generates the random supersingular elliptic module ϕ given by choosing $d = 1 + x$ and computing

$$\begin{aligned} g_\phi &= 3 + 3x + 5x^3 + x^4 + x^5 + x^6 + 6x^7 + 5x^8 + 5x^9, \\ \Delta_\phi &= 4x + 4x^2 + 5x^3 + 3x^4 + 5x^5 + 6x^6 + 5x^7 + 4x^8 + 2x^9. \end{aligned}$$

This means ϕ has complex multiplication by the imaginary quadratic extension $\mathbb{F}_7(x)(\sqrt{x+1})$ of $\mathbb{F}_7(x)$. After computing $\bar{h}_{\phi,f}$, we find $\gamma = \gcd(\bar{h}_{\phi,f}, f) = 1 + 4x + x^2 + 4x^3 + x^4$; this is the product of the factors of f that are supersingular with respect to ϕ. Now, γ does not pass the irreducibility check, so the process is repeated for γ. The next randomly generated supersingular elliptic module is built by choosing $d = 5 + x$ and computing

$$\begin{aligned} g_{\tilde{\phi}} &= 6 + x + 5x^2 + 5x^3, \\ \Delta_{\tilde{\phi}} &= 4 + x + 6x^2 + 4x^3. \end{aligned}$$

We then split γ as $\tilde{\gamma} = \gcd(\bar{h}_{\tilde{\phi},\gamma}, \gamma) = 4 + 6x + x^2$, and $\gamma/\tilde{\gamma} = 2 + 5x + x^2$ which are both irreducible; again, this means that $\tilde{\gamma}$ is supersingular and $\gamma/\tilde{\gamma}$ is ordinary with respect to $\tilde{\phi}$. Finally, $f/\gamma = 2 + 5x + 6x^2 + 3x^3 + 6x^4 + 6x^5 + x^6$ which is also irreducible. The complete factorization of f is then

$$f(x) = (4 + 6x + x^2)(2 + 5x + x^2)(2 + 5x + 6x^2 + 3x^3 + 6x^4 + 6x^5 + x^6).$$

Acknowledgements. We thank the reviewer of a first version of this paper for useful remarks, which led us to this revised version. Doliskani is partially supported by NSERC, CryptoWorks21, and Public Works and Government Services Canada. Narayanan is supported by NSF grant #CCF-1423544, Chris Umans' Simons Foundation Investigator grant and European Union's H2020 Programme under grant agreement number ERC-669891.

References

[1] E. R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Tech. J., 46 (1967), pp. 1853–1849.

[2] W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system. I. The user language*, Journal of Symbolic Computation, 24 (1997), pp. 235–265.

[3] A. Bostan, P. Gaudry, and É. Schost, *Linear recurrences with polynomial coefficients and application to integer factorization and Cartier-Manin operator*, SIAM Journal on Computing, 36 (2007), pp. 1777–1806.

[4] R. P. Brent and H. T. Kung, *Fast algorithms for manipulating formal power series*, Journal of the Association for Computing Machinery, 25 (1978), pp. 581–595.

[5] D. Cantor and H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Mathematics of Computation, 36 (1981), pp. 587–592.

[6] D. G. Cantor and E. Kaltofen, *On fast multiplication of polynomials over arbitrary algebras*, Acta Informatica, 28 (1991), pp. 693–701.

[7] L. Carlitz, *The arithmetic of polynomials in a Galois field*, American Journal of Mathematics, 54 (1932), pp. 39–50.

[8] D. V. Chudnovsky and G. V. Chudnovsky, *Approximations and complex multiplication according to Ramanujan*, in Ramanujan revisited (Urbana-Champaign, Ill., 1987), Academic Press, Boston, MA, 1988, pp. 375–472.

[9] K. Conrad, *Quadratic reciprocity in characteristic 2*. http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/QRchar2.pdf.

[10] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, Journal of Symbolic Computation, 9 (1990), pp. 251–280.

[11] G. Cornelissen, *Deligne's congruence and supersingular reduction of Drinfeld modules*, Archiv der Mathematik, 72 (1999), pp. 346–353.

[12] L. De Feo, J. Doliskani, and É. Schost, *Fast arithmetic for the algebraic closure of finite fields*, in ISSAC'14, ACM, 2014, pp. 122–129.

[13] D. R. Dorman, *On singular moduli for rank 2 Drinfeld modules*, Compositio Mathematica, 80 (1991), pp. 235–256.

[14] M. D. Fried and M. Jarden, *Field arithmetic*, vol. 11 of Ergebnisse der Mathematik und ihrer Grenzgebiete., Springer, third ed., 2008.

[15] J. Gathen and J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, New York, NY, USA, 1999, http://portal.acm.org/citation.cfm?id=304952.

[16] J. Gathen and V. Shoup, *Computing Frobenius maps and factoring polynomials*, Computational Complexity, 2 (1992), pp. 187–224.

[17] E.-U. Gekeler, *On finite Drinfeld modules*, Journal of Algebra, 141 (1991), pp. 187–203.

[18] E.-U. Gekeler, *Frobenius distributions of Drinfeld modules over finite fields*, Transactions of the American Mathematical Society, 360 (2008), pp. 1695–1721.

[19] D. Goss, *Basic Structures of Function Field Arithmetic*, Springer, 1996.

[20] W. B. Hart, *Fast library for number theory: An introduction*, in ICMS'10, Springer, 2010, pp. 88–91. http://flintlib.org.

[21] H. Hasse, *Theorie der relativ-zyklischen algebraischen funktionenkorper insbesondere bei endlichem konstantenkorper*, J. Reine Angew. Math, 172 (1934), pp. 37–44.

[22] X. Huang and V. Y. Pan, *Fast rectangular matrix multiplication and applications*, Journal of Complexity, 14 (1998), pp. 257–299.

[23] E. Kaltofen and V. Shoup, *Subquadratic-time factoring of polynomials over finite fields*, Mathematics of Computation, 67 (1998), pp. 1179–1197.

[24] K. Kedlaya and C. Umans, *Fast polynomial factorization and modular composition*, SIAM Journal on Computing, 40 (2011), pp. 1767–1802.

[25] D. Knuth, *The Art of Computer Programming, Seminumerical Algorithms*, Addison-Wesley, 1997.

[26] F. Le Gall, *Powers of tensors and fast matrix multiplication*, in ISSAC'14, ACM, 2014, pp. 296–303.

[27] H. W. Lenstra, *Factoring integers with elliptic curves*, Annals of Mathematics, 126 (1987), pp. 649–673.

[28] A. K. Narayanan, *Polynomial factorization over finite fields by computing Euler-Poincaré characteristics of Drinfeld modules*. http://arxiv.org/abs/1504.07697, 2015.

[29] A. Panchishkin and I. Potemine, *An algorithm for the factorization of polynomials using elliptic modules*, in Mathematical Institute of AN BSSR, Minsk, Proceedings of the Conference Constructive methods and algorithms in number theory, 1989, p. 117.

[30] M. Rosen, *Number Theory in Function Fields*, Springer, 2002.

[31] T. Scanlon, *Public key cryptosystems based on Drinfeld modules are insecure*, Journal of Cryptology, 14 (2001), pp. 225–230.

[32] A. Schweizer, *On singular and supersingular invariants of drinfeld modules*, Annales de la Facult des sciences de Toulouse : Mathmatiques, 6-2 (1997), pp. 319–364.

[33] V. Shoup, *NTL: A library for doing number theory*. http:/www.shoup.net/ntl.

[34] V. Shoup, *Fast construction of irreducible polynomials over finite fields*, Journal of Symbolic Computation, 17 (1994), pp. 371–391.

[35] V. Strassen, *Einige Resultate über Berechnungskomplexität*, Jber. Deutsch. Math.-Verein., 78 (1976/77), pp. 1–8.

[36] G. J. van der Heiden, *Factoring polynomials over finite fields with Drinfeld modules*, Mathematics of Computation, 73 (2004), pp. 317–322.

[37] D. Y. Y. Yun, *On square-free decomposition algorithms*, in SYMSAC'76, ACM, 1976, pp. 26–38.