

Computation of Hilbert class polynomials and modular polynomials from supersingular elliptic curves

Antonin Leroux

DGA-MI, Bruz, France
IRMAR - UMR 6625, Université de Rennes, France
`antonin.leroux@polytechnique.org`

Abstract. We present several new heuristic algorithms to compute class polynomials and modular polynomials modulo a prime p by revisiting the idea of working with supersingular elliptic curves. The best known algorithms to this date are based on ordinary curves, due to the supposed inefficiency of the supersingular case. While this was true a decade ago, the recent advances in the study of supersingular curves through the Deuring correspondence motivated by isogeny-based cryptography has provided all the tools to perform the necessary tasks efficiently.

Our main ingredients are two new heuristic algorithms to compute the j -invariants of supersingular curves having an endomorphism ring contained in some set of isomorphism class of maximal orders. The first one is derived easily from the existing tools of isogeny-based cryptography, while the second introduces new ideas to perform that task efficiently for a big number of maximal orders at the same time.

For each of the polynomials (Hilbert and modular), we obtain two algorithms. The first one, that we will qualify as *direct*, is based on the computation of a set of well-chosen supersingular j -invariants defined over \mathbb{F}_{p^2} and uses the aforementioned algorithm to translate maximal orders to j -invariants as its main building block. The second one is a CRT algorithm that applies the direct algorithm on a set of small primes and reconstruct the result modulo p with the chinese remainder theorem.

In both cases, the direct algorithm achieves the best known complexity for primes p that are relatively small compared to the discriminant (for the Hilbert case) and to the level (for the modular case).

Our CRT algorithms matches the complexities of the state-of-the-art CRT approach based on ordinary curves, while improving some of the steps, thus opening the possibility to a better practical efficiency.

1 Introduction

Hilbert class polynomials and modular polynomials are central objects in number theory, and their computation have numerous applications. One field where these computations are of particular interest is cryptography. The main applications are to be found in elliptic curve cryptography and pairing-based cryptography,

but we can also mention, more marginally, the recent field of isogeny-based cryptography.

Class polynomials, for instance, play a central role in the CM method, which is the main approach to find ordinary curves with a prescribed number of points over a given finite field (see [AM93,BS07]). This has applications to primality proving with the ECPP method and finding pairing friendly-curves with the Cocks-Pinch method.

Modular polynomials are related to isogenies between elliptic curves. Historically, they play a very important role in the SEA point counting algorithm [E+98,Sch95] which remains one of the main algorithms used in elliptic-curve cryptography to generate cryptographic curves. Moreover, the interest in isogenies have been renewed with the rise of isogeny-based cryptography. While most applications tend to use the more efficient Vélu formulas [Vél71], we can cite a few instances where modular polynomials have been considered. For example, it is used in the CRS key exchange [Cou06,RS06], the very first isogeny-based protocol, and we can also mention the OSIDH construction [CK19].

The goal of this work is to explore theoretical and practical improvements to the best-known algorithms to compute class polynomials and modular polynomials modulo prime numbers through the use of supersingular curves.

Related work. One of the main problems behind the computation of class polynomials and modular polynomials is the huge size of their coefficients over \mathbb{Z} . There exists several algorithms of quasi-linear complexity [Eng09,CH02,Sut11,BLS12], but more often than not, memory is the real bottleneck in the concrete computations of those polynomials. In theory, size is less an issue when the result is needed modulo some prime p , but this is only true in practice if we have a way to skip entirely the computation over \mathbb{Z} , which is not so easy to get.

Nonetheless, Sutherland [Sut11] proved that this could be done for class polynomials by a careful application of the CRT method. The result was later applied to modular polynomials by Bröker, Lauter and Sutherland (BLS) [BLS12]. The main advantage of the CRT method compared to other approaches is the low memory requirement (almost optimal in the size of the final output), and this is why this method has achieved the best practical results.

For both class and modular polynomials, the main tools used in the CRT algorithms from [Sut11,Sut12,BLS12] are /ordinary elliptic curves.

Supersingular curves had been considered in the context of the Hilbert polynomial computation in [BBEL08] and in the context of the modular polynomials computation in [CL05], but these methods were discarded over the time because they were slower than those based on ordinary curves.

The situation has changed with the recent interest on the connection between supersingular curves and quaternion algebras sparked by isogeny-based cryptography.

Since the work of Deuring [Deu41], it is known that endomorphism rings of supersingular curves in characteristic p are isomorphic to maximal orders in the quaternion algebra $\mathcal{B}_{p,\infty}$ ramified at p and infinity, and that, conversely, every such maximal order types arises in this way. This is the first result of what is now

called the *Deuring correspondence*. In this work, we are particularly interested in the task of computing the j -invariants of the (at most 2) supersingular elliptic curves over \mathbb{F}_{p^2} having a given maximal order type as endomorphism ring.

The first concrete effort to realize that task is an algorithm of Cervino [Cer04] to compute the endomorphism rings of all supersingular curves in characteristic p . The complexity of this algorithm is $O(p^{2+\varepsilon})$ and it becomes rapidly impractical. This algorithm was more recently improved by Chevyrev and Galbraith in [CG14] but the complexity is still $O(p^{1.5+\varepsilon})$. As part of cryptanalytic efforts to understand the difficulty of various problems related to the Deuring correspondence, a heuristic algorithm of polynomial complexity in $\log(p)$ was introduced by Eisenträger, Hallgren, Lauter, Morrison and Petit [EHL⁺18]. This algorithm builds upon the previous works of Kohel, Lauter, Petit and Tignol [KLPT14] and Galbraith, Petit and Silva [GPS17].

More concretely, these works prove that an isogeny can be efficiently computed between two supersingular curves of known endomorphism ring by translating the problem over the quaternions with the Deuring correspondence, solving the translated problems over quaternions, before translating back the solution as an isogeny. This can be applied directly to compute the j -invariants of all curves with an endomorphism ring contained in a given maximal order type by using one starting curve E_0 of known endomorphism ring (such a curve can always be computed efficiently with the CM method).

Recently, in [Rob22], Robert introduced new algorithms to compute modular polynomials from supersingular curves, this time using the high-dimensional isogeny techniques recently introduced in the context of isogeny-based cryptography cryptanalysis. Like us, Robert has a direct algorithm and a CRT algorithm. The direct algorithm has a quadratic complexity in the level ℓ , just as our direct algorithm, but the asymptotic complexity is somewhat differently balanced. The CRT algorithm reaches the same complexity as Sutherland's. Robert algorithm have the very nice advantage to have a proven complexity without any heuristic, but they require more memory, in particular the CRT algorithm.

Contributions. Our main contribution is to revisit previous methods to compute Hilbert and modular polynomials from supersingular curves with the recent algorithmic progress on the Deuring correspondence.

Our main sub-routine aims at translating a set of isomorphism class of maximal orders into their corresponding supersingular j -invariants under the Deuring correspondence. We introduce two algorithms, with different performance profiles, to perform that task.

With these new algorithms, we obtain an improvement over the asymptotic complexity of the class and modular polynomials computation in a wide range of primes below some upper-bounds that depend either on the discriminant of the class polynomial or the level of the modular polynomial. Moreover, we show that our new algorithm can also be used in the CRT method to reach the same complexity as ordinary curves, but with possibly better practical efficiency.

1.1 Technical overview

We start by looking at our main-subroutine that consists in the computation of the j -invariants of supersingular elliptic curves corresponding to some set of maximal order isomorphism classes (called maximal order types, see Definition 1). In the rest of this article, unless specified otherwise, a curve is considered to be a supersingular elliptic curve.

Maximal orders to j -invariants. We propose two algorithms dedicated to that task. Let consider that a set \mathfrak{S} of types is given in input, together with some prime p .

Our first algorithm is called `OrdersTojInvariantSmallSet` and it consists merely in a sequential execution of a sub-algorithm from [EHL⁺18] that performs the desired translation for one type of maximal order. When everything is done carefully, it can be executed in $O(\log(p)^{5+\varepsilon})$ under experimentally verified heuristics detailed in [KLPT14] and related to the probability for numbers represented by some quadratic form to be prime. Thus, since `OrdersTojInvariantSmallSet` consists in $\#\mathfrak{S}$ executions of this sub-algorithm, the total heuristic complexity of `OrdersTojInvariantSmallSet` is $O(\#\mathfrak{S} \log(p)^{5+\varepsilon})$. For a generic p and set of maximal order type \mathfrak{S} , we do not know how to do better than that. However, when \mathfrak{S} contains almost all maximal order types (the maximal size being upper-bounded by the number of supersingular curves), it becomes sub-optimal due to the amount of redundant computation performed along the way. In that case, it becomes much more practical to use an algorithm designed to sieve through the entire set of types, only focusing on the ones in \mathfrak{S} when they're met along the way. It requires a bit of care to perform this task in the most efficient manner but it can be done, and this leads to the algorithm `OrdersTojInvariantBigSet` of complexity $O(\#\mathfrak{S} \log(p)^{2+\varepsilon} + p \log(p)^{1+\varepsilon})$. This algorithm requires one heuristic that we detail in Section 3, as Claim 1. It is related to the expansion property of the supersingular isogeny graphs.

We stress that both algorithms are designed to work (and analyzed) for a generic prime p which is why they are so interesting.

The direct algorithms. Our heuristic algorithms `OrdersTojInvariantSmallSet` and `OrdersTojInvariantBigSet` can be used directly to compute the roots of class and modular polynomials modulo p (under the assumption that these roots are supersingular). The method is pretty straightforward: find the maximal order types corresponding to the desired roots, then, compute them with either `OrdersTojInvariantSmallSet` or `OrdersTojInvariantBigSet`. With the complexity we have stated, this is already enough to obtain an asymptotic improvement over existing generic methods when p is not too big (compared to the discriminant or level of the associated class or modular polynomial).

If we write S the "degree" of the polynomial (it is $h(D) = O(\sqrt{|D|} \log(D)^\varepsilon)$ for Hilbert polynomials of discriminant $|D|$ and $O(\ell^2)$ for modular polynomials of level ℓ), then we obtain the following complexity with `OrdersTojInvariantSmallSet`:

$$O(S \log^{5+\varepsilon} p + S \log^{2+\varepsilon} S \log p).$$

With `OrdersToJInvariantBigSet`, the complexity becomes

$$O(S \log^{2+\varepsilon} p + p \log^{1+\varepsilon} p + S \log^{2+\varepsilon} S \log p).$$

In both cases, the latter term comes from the polynomial reconstruction step that must be performed to recover the polynomial from its roots. Note that the size of the output is $O(S \log p)$. In terms of space, the requirement is quasi-optimal in both cases: so $O(S^{1+\varepsilon} \log p)$.

It is clear that the second algorithm will be better when $p = O(S \log(S))$. However, whenever $S = o(p)$ (which is often the case in applications), it will be better to use the variant with `OrdersToJInvariantSmallSet`.

The CRT for class polynomials. Let us take a prime p and a discriminant $D < 0$. We want an efficient algorithm to compute $H_D(X) \bmod p$. Our main algorithm is essentially the same as the one introduced by Belding, Bröker, Enge and Lauter in [BBEL08], and later improved by Sutherland in [Sut11].

Let us write \mathfrak{D} for the quadratic imaginary order of discriminant $D < 0$. We may assume that the factorization of D is known as computing it is negligible compared to the rest of the computation. We define $\mathcal{P}_{\mathfrak{D}}$ to be a set of primes. We write B_D for the bound on the bit-size of the coefficients of H_D over \mathbb{Z} .

Here is how the algorithm works:

1. Select some primes p_1, \dots, p_n in $\mathcal{P}_{\mathfrak{D}}$ with $\prod_{i=1}^n p_i > 2^{B_D}$.
2. Compute a suitable representation of $\text{Cl}(D)$.
3. For each $p_i \in \mathcal{P}_{\mathfrak{D}}$:
 - (a) Compute the coefficients of $H_D \bmod p_i$.
 - (b) Update CRT sums for each coefficient of H_D .
4. Recover the coefficients of $H_D \bmod p$.

The main difference between the several variants of the CRT approach (including ours) lies in the choice of the set $\mathcal{P}_{\mathfrak{D}}$ and in the way to compute $H_D \bmod p_i$. In [BBEL08], different algorithms were proposed to handle the distinct cases of split and inert p_i . In both cases, the $H_D \bmod p_i$ are constructed from their roots. These roots are always j -invariants of elliptic curves in characteristic p_i , but this is where the similarity ends. In the former case, the elliptic curves are ordinary and are defined over \mathbb{F}_{p_i} , whereas in the latter case, we obtain supersingular elliptic curves defined over $\mathbb{F}_{p_i^2}$. The ordinary and supersingular case are very different and the resulting algorithms are also very different.

Sutherland [Sut11] improved the method from [BBEL08] for split primes by a careful choice of the primes and other tricks to improve the computation of $H_D \bmod p_i$.

We improve the method presented in [BBEL08] for non-split primes by making use of the Deuring correspondence with our `OrdersToJInvariantBigSet` algorithm to replace Cervino's algorithm [Cer04].

The CRT for modular polynomials. Let ℓ be a prime distinct from p . We want a CRT approach to compute $\Phi_\ell(X, Y) \bmod p$. It can be done in a very similar fashion to class polynomials by computing $\Phi_\ell \bmod p_i$ for some p_i in a set \mathcal{P}_ℓ and reconstruct the final polynomial via the CRT.

Bröker, Lauter and Sutherland (BLS in the rest of this article) in [BLS12] proposed to use primes of the form $(t^2 - 4v^2D\ell^2)$ with $t, v, D \in \mathbb{N}$ for which there is a very specific volcano structure involving ℓ -isogenies. This structure implies the existence of two distinct sets of ordinary curves defined over \mathbb{F}_{p_i} : the curves with endomorphism ring isomorphic to \mathfrak{D} for some quadratic imaginary order \mathfrak{D} of discriminant D and class number bigger than $\ell + 2$ and the curves with endomorphism ring isomorphic to $\mathbb{Z} + \ell\mathfrak{D}$. Since the latter are ℓ -isogenous to the former, it is possible to recover the full $\Phi_\ell \bmod p_i$ by computing the j -invariants corresponding to these two sets of curves. The volcano structure allows for efficient computation by minimizing the number of ℓ -isogeny computations.

For supersingular curves, the choice of primes is even easier than for class polynomials: we can use any primes p_i that is big enough. As long as the number of supersingular curves is bigger than $\ell + 2$ we will be able to recover the full modular polynomial. This idea has already been considered by Charles and Lauter in 2005 [CL05] but in a rather direct way (where each of the ℓ -isogeny involved is computed using the Vélú formulas).

We prove that using the Deuring correspondence and `OrdersTojInvariantBigSet`, we can avoid entirely any ℓ -isogeny computation and minimize the cost of elliptic curve operations.

Generic improvements to the CRT method. There are several ways to improve the CRT method in practical applications. First, alternative class polynomials and modular functions (with smaller height bounds) can be used instead of the standard Hilbert class polynomial and modular polynomials for the same practical purpose. This was the focus of the paper [ES10].

Second, for a number of applications such as the CM method and the SEA point counting algorithm, computing these polynomials is actually not necessary. What is really needed is the ability to evaluate them. Sutherland showed [Sut12,Sut13] that it was possible to do better than compute-then-evaluate in both cases, providing, in particular, an additional improvement in terms of memory requirement for those applications.

Using supersingular curves rather than ordinary ones should not prevent from applying all these practical improvements. For clarity's sake we focus on the simpler computation of the standard polynomials and leave to the reader the task of adapting these improvements to our new setting which should not be too daunting.

Organisation of the article. The rest of this paper is organized as follows: in Section 2, we introduce some background on isogenies, quaternion algebras and the Deuring correspondence. Then, in Section 3, we introduce our main new algorithm to compute efficiently j -invariants corresponding to maximal order types. In Section 4, we explain in details how this algorithm can be applied to

the computation of class polynomial with the CRT method. In Section 5, we do the same for modular polynomials.

Acknowledgement. We thank Andrew Sutherland and anonymous reviewers for very useful feedback on this work.

2 Background material

2.1 Notations

Basic complexities. We write $M_{\mathbb{Z}}(b)$ for the cost of multiplying two integers of less than b bits. For asymptotic complexities we consider $M_{\mathbb{Z}}(b) = O(b^{1+\varepsilon})$. For instance, this covers the complexity of all arithmetic operations in a finite field \mathbb{F}_p of characteristic p of less than b bits.

Similarly, we write $M_{\mathbb{P}}(b)$ for the cost (in terms of arithmetic operations over k) of multiplying two polynomials of degree smaller than b over a base field k . Depending on the size of b we will either use $M_{\mathbb{P}}(b) = O(b \log(b)^{1+\varepsilon})$ or $O(b^{1+\varepsilon})$.

Finally, the cost of fast interpolation algorithm for a polynomial of degree b is $O(M_{\mathbb{P}}(b) \log(b))$.

2.2 Elliptic curves, quaternion algebras and the Deuring correspondence

More precise references on the topics covered in this section are: the book of Silverman [Sho94] for elliptic curves and isogenies, the book of John Voight [Voi18] on quaternion algebras and theoretical aspects of the Deuring correspondence, the thesis of Antonin Leroux [Ler22] for the algorithmic aspects of the Deuring correspondence.

Supersingular elliptic curves and isogenies. An *isogeny* $\varphi : E_1 \rightarrow E_2$ is a non-constant morphism sending the identity of E_1 to that of E_2 . The degree of an isogeny is its degree as a rational map (see [HS09] for more details). When the degree $\deg(\varphi) = d$ is coprime to p , the isogeny is necessarily *separable* and $d = \#\ker \varphi$. An isogeny is said to be *cyclic* when its kernel is a cyclic group. The Vélú formulas [Vél71] can be used to compute any cyclic isogeny from its kernel. For any $\varphi : E_1 \rightarrow E_2$, there exists a unique dual isogeny $\hat{\varphi} : E_2 \rightarrow E_1$, satisfying $\varphi \circ \hat{\varphi} = [\deg(\varphi)]$.

Endomorphism ring. An isogeny from a curve E to itself is an *endomorphism*. The set $\text{End}(E)$ of all endomorphisms of E forms a ring under addition and composition. For elliptic curves defined over a finite field \mathbb{F}_q , $\text{End}(E)$ is isomorphic either to an order of a quadratic imaginary field or a maximal order in a quaternion algebra. In the first case, the curve is said to be *ordinary* and otherwise *supersingular*. We focus on the supersingular case in this article. Every supersingular elliptic curve defined over a field of characteristic p admits an isomorphic

model over \mathbb{F}_{p^2} . It implies that there only a finite number of isomorphism class of supersingular elliptic curves. The Frobenius over \mathbb{F}_p is the only inseparable isogeny between supersingular curves and it has degree p . We write $\pi : E \rightarrow E^p$. For any supersingular curve E , the property $\text{End}(E) \cong \text{End}(E^p)$ is satisfied but we have $E \cong E^p$ if and only if E has an isomorphic model over \mathbb{F}_p .

Quaternion algebras. For $a, b \in \mathbb{Q}^*$ we denote by $H(a, b) = \mathbb{Q} + i\mathbb{Q} + j\mathbb{Q} + k\mathbb{Q}$ the quaternion algebra over \mathbb{Q} with basis $1, i, j, k$ such that $i^2 = a$, $j^2 = b$ and $k = ij = -ji$. Every quaternion algebra has a canonical involution that sends an element $\alpha = a_1 + a_2i + a_3j + a_4k$ to its conjugate $\bar{\alpha} = a_1 - a_2i - a_3j - a_4k$. We define the *reduced trace* and the *reduced norm* by $\text{tr}(\alpha) = \alpha + \bar{\alpha}$ and $n(\alpha) = \alpha\bar{\alpha}$.

Orders and ideals. A *fractional ideal* I of a quaternion algebra \mathcal{B} is a \mathbb{Z} -lattice of rank four contained in \mathcal{B} . We denote by $n(I)$ the *norm* of I , defined as the \mathbb{Z} -module generated by the reduced norms of the elements of I .

An order \mathcal{O} is a subring of \mathcal{B} that is also a fractional ideal. Elements of an order \mathcal{O} have reduced norm and trace in \mathbb{Z} . An order is called *maximal* when it is not contained in any other larger order. A suborder \mathfrak{O} of \mathcal{O} is an order of rank 4 contained in \mathcal{O} .

In this work, we will work with isomorphism classes of maximal orders in some quaternion algebra \mathcal{B} and this is why we introduce the notion of type.

Definition 1. *The type of an order \mathcal{O} written $\text{Typ } \mathcal{O}$ is the isomorphism class of \mathcal{O} .*

The left order of a fractional ideal is defined as $\mathcal{O}_L(I) = \{\alpha \in \mathcal{B}_{p,\infty} \mid \alpha I \subset I\}$ and similarly for the right order $\mathcal{O}_R(I)$. A fractional ideal is *integral* if it is contained in its left order, or equivalently in its right order. An integral ideal is *primitive* if it is not the scalar multiple of another integral ideal. We refer to integral primitive ideals hereafter as ideals.

The product IJ of ideals I and J satisfying $\mathcal{O}_R(I) = \mathcal{O}_L(J)$ is the ideal generated by the products of pairs in $I \times J$. It follows that IJ is also an (integral) ideal and $\mathcal{O}_L(IJ) = \mathcal{O}_L(I)$ and $\mathcal{O}_R(IJ) = \mathcal{O}_R(J)$. The ideal norm is multiplicative with respect to ideal products. An ideal I is invertible if there exists another ideal I^{-1} verifying $II^{-1} = \mathcal{O}_L(I) = \mathcal{O}_R(I^{-1})$ and $I^{-1}I = \mathcal{O}_R(I) = \mathcal{O}_L(I^{-1})$. The conjugate of an ideal \bar{I} is the set of conjugates of elements of I , which is an ideal satisfying $I\bar{I} = n(I)\mathcal{O}_L(I)$ and $\bar{I}I = n(I)\mathcal{O}_R(I)$.

We define an equivalence on orders by conjugacy and on left \mathcal{O} -ideals by right scalar multiplication. Two orders \mathcal{O}_1 and \mathcal{O}_2 are equivalent if there is an element $\beta \in \mathcal{B}^*$ such that $\beta\mathcal{O}_1 = \mathcal{O}_2\beta$. Two left \mathcal{O} -ideals I and J are equivalent if there exists $\beta \in \mathcal{B}^*$, such that $I = J\beta$. If the latter holds, then it follows that $\mathcal{O}_R(I)$ and $\mathcal{O}_R(J)$ are equivalent since $\beta\mathcal{O}_R(I) = \mathcal{O}_R(J)\beta$. For a given \mathcal{O} , this defines equivalence classes of left \mathcal{O} -ideals, and we denote the set of such classes by $\text{Cl}(\mathcal{O})$.

The *Deuring correspondence* is an equivalence of categories between isogenies of supersingular elliptic curves and the left ideals over maximal order \mathcal{O} of $\mathcal{B}_{p,\infty}$, the unique quaternion algebra ramified at p and ∞ , inducing a bijection between conjugacy classes of supersingular j -invariants and maximal orders (up to equivalence) [Koh96]. Moreover, this bijection is explicitly constructed as $E \rightarrow \text{End}(E)$. Hence, given a supersingular curve E_0 with endomorphism ring \mathcal{O}_0 , the pair (E_1, φ) , where E_1 is another supersingular elliptic curve and $\varphi : E_0 \rightarrow E_1$ is an isogeny, is sent to a left integral \mathcal{O}_0 -ideal. The right order of this ideal is isomorphic to $\text{End}(E_1)$. One way of realizing this correspondence is obtained through the kernel ideals defined in [Wat69]. Given an integral left- \mathcal{O}_0 -ideal I , we define the kernel of I as the subgroup

$$E_0[I] = \{P \in E_0(\overline{\mathbb{F}}_{p^2}) : \alpha(P) = 0 \text{ for all } \alpha \in I\}.$$

To I , we associate the isogeny

$$\varphi_I : E_0 \rightarrow E_0/E_0[I].$$

Conversely, given an isogeny φ , the corresponding *kernel ideal* is

$$I_\varphi = \{\alpha \in \mathcal{O}_0 : \alpha(P) = 0 \text{ for all } P \in \ker(\varphi)\}.$$

In Table 1, we recall the main features of the Deuring correspondence.

Supersingular j -invariants over \mathbb{F}_{p^2}	Maximal orders in $\mathcal{B}_{p,\infty}$
$j(E)$ (up to Galois conjugacy)	$\mathcal{O} \cong \text{End}(E)$ (up to isomorphism)
(E_1, φ) with $\varphi : E \rightarrow E_1$	I_φ integral left \mathcal{O} -ideal and right \mathcal{O}_1 -ideal
$\theta \in \text{End}(E_0)$	Principal ideal $\mathcal{O}\theta$
$\deg(\varphi)$	$n(I_\varphi)$
$\hat{\varphi}$	\hat{I}_φ
$\varphi : E \rightarrow E_1, \psi : E \rightarrow E_1$	Equivalent Ideals $I_\varphi \sim I_\psi$
Supersingular j -invariants over \mathbb{F}_{p^2}	$\text{Cl}(\mathcal{O})$
$\tau \circ \rho : E \rightarrow E_1 \rightarrow E_2$	$I_{\tau \circ \rho} = I_\rho \cdot I_\tau$

Table 1: The Deuring correspondence, a summary from [DFKL⁺20].

Effective Deuring correspondence After establishing the nice theoretical results of the Deuring correspondence, it is natural to ask if we can obtain efficient algorithms to perform the translation between the two sides of our correspondence. This trend of work was started by Kohel, Lauter, Petit and Tignol in [KLPT14], and developed by Galbraith, Petit and Silva in [GPS17]. In [EHL⁺18], Eisentrager, Haller, Lauter, Petit and Morrison provided the first complete picture of the situation (at least heuristically). It turns out that if we start from the quaternion side (either as a maximal order or an ideal), there are polynomial-time algorithms to compute the corresponding element (j -invariant, or isogeny).

In particular, Eisentrager et al. introduced a heuristic polynomial-time algorithm that computes the j -invariant corresponding to a maximal order type given in input. Henceforth, we call this algorithm `SingleOrderTojInvariant()`. It will be a crucial building block in one of our algorithms.

3 Computing j -invariants corresponding to maximal orders.

Let us fix some prime number p .

In this section, we introduce two algorithms to compute the j -invariants of supersingular curves over \mathbb{F}_{p^2} corresponding to a set \mathfrak{S} of maximal order types in $\mathcal{B}_{p,\infty}$. By the Deuring correspondence, we know that each maximal order type in $\mathcal{B}_{p,\infty}$ corresponds to one or two j -invariants of supersingular curve over \mathbb{F}_{p^2} .

We will explain in Section 3.1 how to represent efficiently maximal order types as elements in some set \mathcal{H} . Concretely, the input \mathfrak{S} to our algorithms will be given as some subset of \mathcal{H} .

Our two algorithms presented in Sections 3.2 and 3.3 target two opposite situations with respect to the relative size of p and \mathfrak{S} . The first algorithm is called `OrdersTojInvariantSmallSet`. As the name suggests, it targets the case where $\#\mathfrak{S}/p$ is "small". It is a quite direct application of standard results on the effective Deuring correspondence, and handles each maximal order independently. The asymptotic complexity is $O(\#\mathfrak{S} \log p^{5+\varepsilon})$ and works for any prime p and set \mathfrak{S} .

The second algorithm is called `OrdersTojInvariantBigSet` and it is more involved in both design and analysis. It targets the case where \mathfrak{S} is made of a significant portion of all $O(p)$ possible types and is based on the idea that since \mathfrak{S} is big enough, the strategy that consists in going through the entire supersingular isogeny graph, collecting the j -invariants we want along the way, is quite optimal. Its complexity is $O(\#\mathfrak{S} \log p^{2+\varepsilon} + p \log p^{1+\varepsilon})$. Hence, the cutoff between the two methods will be for some \mathfrak{S} with $\#\mathfrak{S} = \Theta(p/\log p^{2+\varepsilon})$. Note that our second algorithm will be optimal when $p/\#\mathfrak{S} = \Theta(\log p^{1+\varepsilon})$.

3.1 Hashing to maximal order types

One of the important point for making our algorithms practical is to have a good way to handle sets of maximal order types and test if a type belong in some set of types.

For any maximal order \mathcal{O} , we will represent `Typ \mathcal{O}` by an invariant $H(\mathcal{O})$. The purpose of this section is to introduce an efficiently computable invariant $H(\mathcal{O})$ and the corresponding function H .

To derive an invariant for an isomorphism classes of lattices, it is quite natural to look at the smallest elements of that lattice. This idea was introduced by Chevyrev and Galbraith [CG14] in a related context, and their result was recently strengthened by Goren and Love [GL23]. Let us take a maximal order \mathcal{O} . It can be shown (see [CG14] and [GL23, Theorem 1.4]) that if x_1, x_2, x_3 realize the

successive minimas of the Gross lattice $\mathcal{O}^T = \{2x - \text{tr}(x) \mid x \in \mathcal{O}\}$, then the values n_1, n_2, n_3 where $n_i = \mathbf{n}(x_i)$ for $i = 1, 2, 3$ determine uniquely the maximal order \mathcal{O} .

This is enough to obtain an invariant of size $O(\log p)$ as it can be shown that $\log(n_i) = O(\log p)$ for all $i \in \{1, 2, 3\}$. If needed, for compactness, one can then apply some kind of hash function $h : \{0, 1\}^* \rightarrow \mathcal{H}$ where \mathcal{H} is big enough to make the probability of a collision negligible over all maximal order types in $\mathcal{B}_{p, \infty}$.

For a generic statement, we take an arbitrary function h (which might be the identity) and assume its computational cost is negligible. Then, we define H as:

1. Compute \mathcal{O}^T .
2. Compute the three successive minimas x_1, x_2, x_3 of \mathcal{O}^T .
3. Output $H(\mathcal{O}) = h(\mathbf{n}(x_1), \mathbf{n}(x_2), \mathbf{n}(x_3))$.

Proposition 1. *The hash function H presented above can be computed in $O(\log(X)^{1+\epsilon})$ when all the coefficients in the decomposition of the basis of \mathcal{O} over $\langle 1, i, j, k \rangle$ are smaller than X .*

Proof. This can be achieved by computing the successive minimas of \mathcal{O}^T with the algorithm to reduce ternary quadratic form of [ER01], or the \tilde{L}^1 algorithm from [NSV11] to perform lattice reduction in small dimension.

3.2 Matching maximal orders and j -invariant: a generic algorithm.

Let us fix a prime p . We assume that a function H as introduced in Section 3.1 is defined and we assume that the underlying hash function h is such that there is no collisions over all maximal order types in $\mathcal{B}_{p, \infty}$.

The algorithm `OrdersTojInvariantSmallSet` is easily derived from the ideal-to-isogeny algorithm described as [EHL⁺18, Algorithm 12] and whose main building blocks were first introduced in [GPS17]. We provide a somewhat detailed description of this algorithm below. We refer to [GPS17, EPSV23] for a description of the main algorithmic building blocks.

1. Compute a curve E_0 and a maximal order \mathcal{O}_0 with $\text{End}(E_0) \cong \mathcal{O}_0$.
2. Set $M = \{\}$.
3. For each $\mathcal{O} \in \mathfrak{S}$ do:
 4. Compute an ideal I of powersmooth norm connecting \mathcal{O}_0 and \mathcal{O}
 5. Translate that ideal into the corresponding isogeny φ_I .
 6. Compute the j -invariant $j_{\mathcal{O}}$ of the codomain of φ_I .
 7. $M = M \cup \{H(\mathcal{O}), j_{\mathcal{O}}\}$.
8. Return M .

Under some heuristic assumptions regarding the distribution of numbers represented by quadratic forms (see [KLPT14]), the complexity of this algorithm is $O(\#\mathfrak{S} \log p^{5+\epsilon})$.

For a concrete implementation, we propose to use the approach described in [EPSV23].

3.3 Maximal orders to j -inv: a faster algorithm when $\#\mathfrak{S} \approx p$.

`OrdersToJInvariantSmallSet` treats independently each order in \mathfrak{S} . This is fine when \mathfrak{S} is quite small compared to the number of maximal order types in $\mathcal{B}_{p,\infty}$, but it becomes less and less optimal as $\#\mathfrak{S}/p$ increases. Indeed, we end up redoing a lot of computations in that manner.

To overcome this problem when the ratio $\#\mathfrak{S}/p$ increases, we can try to mutualize as much computation as possible by using an approach that explores the entire isogeny graph and only targets the specific elements of \mathfrak{S} along the way. Since, our exploration of the isogeny graph is completely generic, we obtain an algorithm with a much better amortized cost per maximal order type.

More concretely, our idea is the following: take a smooth degree L such that all supersingular curves are L -isogenous to some starting curve E_0 of endomorphism ring in $\text{Typ } \mathcal{O}_0$ for some maximal order \mathcal{O}_0 . Compute all the \mathcal{O}_0 -ideals of norm L and their right orders and select the ones contained in the set \mathfrak{S} . Then, enumerate efficiently through all the corresponding isogenies of degree L and collect the j -invariants of the codomains. Since quaternion operations cost less, we will do the exhaustive part over the quaternions, while minimizing the cost of elliptic curve operations by "selecting" the L -isogenies than we cannot avoid to compute.

Note that we do the ideal and isogeny phases in a simultaneous manner in `OrdersToJInvariantBigSet`. The good complexity we obtain will come from the care we take in computing all the required isogenies in the most efficient way possible, and in particular to avoid as many useless isogeny computations as possible. For that, the choice of L will be very important. In particular, if $L = L_1L_2$, by factorization of isogenies, we can compute all relevant L -isogenies by computing all L_1 -isogenies and only a subset of all the $O(L_1L_2)$ L_2 -isogenies. This subset obviously depends on \mathfrak{S} and the L_1L_2 -isogeny computations account for the $\#\mathfrak{S} \log p^2$ terms in the complexity (because we will choose L_2 to be smooth). The $p \log p$ covers the costs of the quaternion operations (which does not depend on \mathfrak{S} since we cover all maximal order types) and L_1 isogeny computations.

Before describing the algorithm in itself, we need to provide several properties and one heuristic claim that are going to be crucial for analyzing the algorithm complexity and proving its correctness.

Preliminary results and a heuristic assumption. Our first results target the degree L of the isogenies that we will use. As we explained above, this degree is crucial for optimizing the algorithm. To enable the fast computation of a lot of L -isogenies at the same time, we need it to be power-smooth (for efficient application of the Vélu formulas), but with coprime factors that are not too small.

Let us write $\Phi(N)$ the number of cyclic isogenies of degree N for any $N \in \mathbb{N}$.

Lemma 1. *There exists a bound B and constants $B_0, C_0, 1 < C_1 < C_2$ such that for every number $N > B$, there exists a value $n < B_0 \log(N)$ and integers $(e_i)_{1 \leq i \leq n}$ such that, if we define ℓ_i to be the $(n - i + 1)$ -th smallest prime and*

$L_i = \ell_i^{e_i}$ then we have $\sqrt{C_0 \log(N)} \leq \Phi(L_i) < C_0 \log(N)$ for all $1 \leq i \leq n-1$, $(C_0/2) \log(N) \leq \Phi(L_n) < C_0 \log(N)$ and $C_1 N \leq \prod_{i=1}^n \Phi(L_i) < C_2 N$.

Proof. We remind that when ℓ is a prime number and $e \geq 1$ is an integer, $\Phi(\ell^e) = \ell^{e-1}(1 + \ell)$.

Let us take $C_0, 1 < C_1 < C_2$ three constants. We leave these constants unspecified for now. We start our reasoning without assuming anything on those constants, and we will encounter conditions on them during the proof. At the end, we will verify that these constraints can all be satisfied.

We also take a bound B that we assume to be "big enough" for various asymptotic inequalities to be verified. Let us finally take any integer $N > B$.

Let us construct two integers n, A_n recursively from $A_0 = 1$ in the following manner: if $A_i \geq C_1 N$, then set $n = i$, otherwise let $A_{i+1} = \Phi(\lambda_{i+1}) A_i$ with $\lambda_{i+1} = \ell_{n-i}^{e_{n-i}}$ where each ℓ_i is the $n-i+1$ -th prime and e_i is defined to be 1 if $\Phi(\ell_i) \geq C_0 \log(N)$ or the biggest exponent such that $\Phi(\lambda_{i+1}) < C_0 \log(N)$.

It is clear that this recursive algorithm always terminates and so we can get two integers n, A_n in that manner for any N . Moreover, we have $C_0/2 \log(N) \leq \Phi(\lambda_1)$ since $\ell_1 = 2$ and $\sqrt{C_0 \log(N)} \leq \Phi(\lambda_i)$ for all other i (because if $\ell_i \leq \Phi(\lambda_i) < \sqrt{C_0 \log(N)}$, then $\ell_{n-i} \Phi(\lambda_i) = \Phi(\ell_{n-i} \lambda_i) \leq C_0 \log(N)$ and this is impossible by construction of the exponent e_{n-i}).

Then, there are two possibilities: either $A_n < C_2 N$ or $A_n \geq C_2 N$.

First case: $A_n < C_2 N$. Let us set $L_i = \lambda_{n-i+1}$ for $1 \leq i \leq n$, we will show that L_1, \dots, L_n satisfies all the required properties.

In fact, the only thing that remains to be verified is that $\Phi(L_i) < C_0 \log(N)$ for $1 \leq i \leq n-1$. By construction of the L_i , it is clear that it suffices to prove that the n -th prime (the divisor of L_1) is smaller than $C_0 \log(N) - 1$.

By taking logarithms of the inequality $\sqrt{C_0 \log(N)}^n \leq A_n < C_2 N$, we obtain $(n/2)(\log(C_0) + \log \log(N)) < A_n < \log(C_2) + \log(N)$. From there, it follows (since B is big enough) that $n < 3 \log(N) / \log \log(N)$.

From asymptotic results on the size of the n -th prime (see for instance [RS62, 3.13]), we get that we can assume $\ell_1 < 2n \log(n) < 6 \log(N)$. Thus, if $C_0 \geq 7$, the result is proven.

Second case: $A_n \geq C_2 N$. Since A_n is too big, we will try to divide A_n by a factor δ . Since N is big enough, we can assume $n \geq 4$. We don't want to modify λ_1 since the bound is tight, but we can divide λ_2, λ_3 by some powers of $\ell_{n-1} = 3, \ell_{n-2} = 5$. The idea is that we can remove up to $O(\sqrt{\log(N)})$ from λ_2, λ_3 .

Since we have $A_{n-1} < C_1 N$, by the same reasoning as before we get $n < 1 + 3 \log(N) / \log \log(N)$. Hence, with $\ell_1 < 2n \log(n)$ we get $\ell_1 < 7 \log(N)$.

Since, $\Phi(\lambda_n)$ is either smaller than $C_0 \log(N)$ or equal to $\ell_1 + 1$, we deduce that $\Phi(\lambda_n) < \max(8, C_0) \log(N)$ and it suffices to choose $C_0 \geq 8$ to ensure that $\Phi(\lambda_n) < C_0 \log(N)$.

From there, we deduce $\Lambda_n = \Lambda_{n-1}\Phi(\lambda_n) < C_0C_1N \log(N)$, from which we get

$$C_1N \leq \Lambda_n < C_0C_1N \log(N). \quad (3.1)$$

Now, we want to find $\delta|\Lambda_n$ such that $C_1N \leq \Lambda_n/\delta < C_2N$. We start by using λ_2 . Let us take δ_2 as the biggest divisor of $\Phi(\lambda_2)$ such that

$$\delta_2 < \min(\Lambda_n/(C_1N), \sqrt{C_0 \log(N)}/3).$$

If we assume $C_2 > 3C_1$, then we can clearly take $\delta_2 \geq 3$ without having $\Lambda_n/\delta_2 \leq C_1N$.

If $\Lambda_n/\delta_2 < C_2N$, then we are done with $L_{n-1} = \lambda_2/\delta_2$ and $L_i = \lambda_{n-i+1}$ for all other $1 \leq i \leq n$, as we have $C_1N \leq \Lambda_n/\delta_2 < C_2N$ and the bound $\Phi(\lambda_2)/\delta_2 > \sqrt{C_0 \log(N)}$ is satisfied by choice of δ_2 and the fact that $\Phi(\lambda_2) > C_0 \log(N)/3$.

If not and $\Lambda_n/\delta_2 \geq C_2N$, we will now try to remove a divisor of λ_3 . Before that, it is useful to try to upper-bound Λ_n/δ_2 more precisely.

We have $\delta_2 > \min(\Lambda_n/(C_1N), \sqrt{C_0 \log(N)}/3)/3$ (otherwise we could have multiplied δ_2 by 3). If the inequality $\Lambda_n/(C_1N) < \sqrt{C_0 \log(N)}/3$ were to be true, then we would have $\delta_2 > \Lambda_n/(3C_1N)$ and so $\Lambda_n/\delta_2 < 3C_1N$. This is smaller than C_2N by our assumption on C_1 and C_2 , and so this is a contradiction. Thus we must have $\Lambda_n/(C_1N) \geq \sqrt{C_0 \log(N)}/3$ and we can get the lower-bound $\delta_2 \leq \sqrt{C_0 \log(N)}/9$. If we plug that into Eq. (3.1), we obtain the following

$$C_1N \leq \Lambda/\delta_2 < 9\sqrt{C_0}C_1N\sqrt{\log(N)}. \quad (3.2)$$

Now, we define δ_3 as the biggest divisor of $\Phi(\lambda_3)$ such that

$$\delta_3 < \min(\Lambda_n/(C_1\delta_2N), \sqrt{C_0 \log(N)}/\ell_3).$$

Then, following the same reasoning as for δ_2 , we get that if we assume further $5C_1 < C_2$, then we can take a non-trivial δ_3 . Furthermore, we have that either $\Lambda_n/(\delta_2\delta_3) < C_2N$ and we are done (for the same reasons as before), or we must have $\Lambda_n/(\delta_2C_1N) \geq \sqrt{C_0 \log(N)}/\ell_3$.

In that case, we can plug this in Eq. (3.2) to get

$$C_1N \leq \frac{\Lambda_n}{\delta_2\delta_3} < 225C_1N. \quad (3.3)$$

If we assume that $C_2 > 225C_1$ this is impossible and so we are done.

In summary, suitable integers L_1, \dots, L_n can be found for any big enough N assuming that the constants C_0, C_1, C_2 satisfies:

- $C_0 > 7$.
- $C_2 > 3C_1$.
- $C_2 > 5C_1$.
- $C_2 > 225C_1$.

This system of inequalities can clearly be satisfied and so this proves the result.

Definition 2. Given any integer $N > B$, the integers L_1, \dots, L_n as defined in Lemma 1 is called a **good-degree basis** for N .

We derive the following result that will be useful in our analysis.

Proposition 2. Take an integer $N > B$ with good degree-basis L_1, L_2, \dots, L_n with $n > 7$. Then, there exists a constant C_3 such that

$$\sum_{i=1}^{n-6} (n-i) \prod_{j=1}^i \Phi(L_j) < C_3 \frac{N}{\log(N)^2} \quad (3.4)$$

Proof. To prove the result, we are going to use the equalities $\sum_{i=0}^m X^i = \frac{X^{m+1}-1}{X-1}$ and $\sum_{i=1}^m iX^i = X \left(\frac{mX^{m+1} - (m+1)X^m + 1}{(X-1)^2} \right)$ for any m and X . By our definition of a degree-basis in Lemma 1, we have that $\Phi(L_j) \geq \sqrt{C_0 \log(N)}$ for all $1 \leq j \leq n$. Thus, $\prod_{j=1}^i \Phi(L_j) < C_2 N / \prod_{j=i+1}^n \Phi(L_j) < C_2 N / (\sqrt{C_0 \log(N)})^{n-i}$. Let us take $X = 1/\sqrt{C_0 \log(N)}$. We have $\sum_{i=1}^{n-6} (n-i) \prod_{j=1}^i \Phi(L_j) < C_2 N \sum_{i=1}^{n-6} (n-i) X^{n-i}$. Then, we have

$\sum_{i=1}^{n-6} (n-i) X^{n-i} = \sum_{i=6}^{n-1} i X^i = X^6 \sum_{i=0}^{n-7} (i+6) X^i$. Then, we apply our two equalities to get

$$\begin{aligned} \sum_{i=1}^{n-6} (n-i) X^{n-i} &= X^6 \left(6 \frac{X^{n-6} - 1}{X-1} + \frac{(n-7)X^{n-5} - (n-6)X^{n-6} + X}{(X-1)^2} \right) \\ &= X^6 \frac{(n-1)X^{n-5} - (n-12)X^{n-6} - 5X + 6}{(X-1)^2}. \end{aligned}$$

Since, asymptotically, we will have $X < 1$ while n will increase, we have that the leading term in the numerator of the fraction is 1, and so there exists C'_3 such that $\sum_{i=1}^{n-6} (n-i) X^{n-i} \leq C'_3 X^4$. Thus, $\sum_{i=1}^{n-6} (n-i) \prod_{j=1}^i \Phi(L_j) < \frac{C_2 C'_3}{C_0^2} \frac{N}{\log(N)^2}$. \square

Our heuristic claim is about the size of L required to meet the condition that all supersingular curves are L -isogenous to some curve E_0 . We rewrite this under the Deuring correspondence as a condition on maximal orders and ideals.

Claim 1 There exists a constant C_4 such that for any prime p and maximal order \mathcal{O}_0 in $\mathcal{B}_{p,\infty}$, given any number $N > pC_4$, every maximal order type in $\mathcal{B}_{p,\infty}$ is obtained as the type of the right order of a left integral \mathcal{O}_0 -ideal of norm N .

Remark 1. Claim 1 is consistent with experiments regarding the diameter of the graph of supersingular 2-isogenies made in [ACNL⁺21]. We also made some small experiments that seems to be consistent with that idea.

In case our claim fails, it is possible to use several starting curve E_0 to decrease the probability of missing some curve. If this is still not enough, and

a few types are not obtained in this manner, it is always possible to apply `SingleOrderTojInvariant` to compute the remaining j -invariants without damaging too much the complexity.

For a given input p to `OrdersTojInvariantBigSet`, we assume the knowledge of a good degree-basis L_1, \dots, L_n (see Definition 2) of C_4p (where C_4 is the constant in Claim 1). In `OrdersTojInvariantBigSet`, we will use the L_i torsion points for all i . For supersingular curves, these points can always be defined over an extension $\mathbb{F}_{p^{m_i}}$. We have the following lemma to bound the value of all the m_i from L_i .

Lemma 2. *Let p be a prime number and E_0 a supersingular curve over \mathbb{F}_{p^2} . For any integer N , coprime with p , there exists $\hat{E}_0 \cong E_0$ over $\overline{\mathbb{F}}_p$ such that the torsion subgroup $\hat{E}_0[N]$ is defined over an extension \mathbb{F}_{p^m} of even degree $m \leq N$ over \mathbb{F}_p .*

Algorithm 1 OrdersToJInvariantBigSet

Input: A prime p , and a set of maximal order types in $\mathcal{B}_{p,\infty}$.

Output: The set of j -invariants corresponding to the maximal orders of \mathfrak{S} .

- 1: Establish L_1, \dots, L_n , a good-degree basis for the number of supersingular curves.
 - 2: Compute a supersingular curve E_0 over \mathbb{F}_{p^2} with known endomorphism ring \mathcal{O}_0 .
 - 3: Compute $I_0 = \mathcal{O}_0\langle\alpha_0, L\rangle$ one left integral \mathcal{O}_0 -ideal of norm $L := \prod_{i=1}^n L_i$.
 - 4: Find $\alpha \in \text{End}(E_0)$ such that $\gcd(\mathbf{n}(x+y\alpha), L) = 1$ for all x, y with $\gcd(x, y, L) = 1$.

 - 5: **for** $i \in [1, \dots, n]$ **do**
 - 6: Compute a basis $P_{0,i}, Q_{0,i}$ of $E_0[L_i]$ over $\mathbb{F}_{p^{m_i}}$.
 - 7: Compute the generator $R_{0,i}$ of $E[I_0 + \mathcal{O}_0 L_i]$ from P_i, Q_i .
 - 8: Compute $S_{0,i} = \alpha(R_{0,i})$.
 - 9: **end for**
 - 10: Set $\text{List} = [\{E_0, \mathcal{O}_0\langle 1 \rangle, [R_{0,1}, \dots, R_{0,n}], [S_{0,1}, \dots, S_{0,n}]\}]$.
 - 11: Set $M = \{\}$ and $m = 0$.
 - 12: **if** $H(\mathcal{O}_0) \in \mathfrak{S}$ **then**
 - 13: $M = M \cup \{(H(\mathcal{O}_0), j(E_0))\}$ and $m = m + 1$.
 - 14: **end if**
 - 15: **for** $i \in [1, \dots, n]$ **do**
 - 16: $\text{NewList} = []$.
 - 17: **for** $x \in \text{List}$ **do**
 - 18: Parse x as $E, I, [R_i, \dots, R_n], [S_i, \dots, S_n]$.
 - 19: **for** all cyclic subgroups $\langle CR_i + [D]S_i \rangle$ of order L_i in $\langle R_i, S_i \rangle$ **do**
 - 20: Compute $P := [C]R_i + [D]S_i$.
 - 21: Compute J the ideal $\mathcal{O}_0\langle\alpha_0(C + D\bar{\alpha}), L_i\rangle$.
 - 22: Set $K := I \cap J$.
 - 23: Set $\text{List}_1 = [], \text{List}_2 = []$.
 - 24: Let $\mathcal{O} = \mathcal{O}_R(K)$.
 - 25: **if** $i < n$ **then**
 - 26: Compute $\varphi : E \rightarrow E/\langle P \rangle$.
 - 27: Compute $\text{List}_1 = [\varphi(R_{i+1}), \dots, \varphi(R_n)]$.
 - 28: Compute $\text{List}_2 = [\varphi(S_{i+1}), \dots, \varphi(S_n)]$.
 - 29: **end if**
 - 30: **if** $H(\mathcal{O}) \in \mathfrak{S}$ and not contained in M already **then**
 - 31: **if** $i = n$ **then**
 - 32: Compute $\varphi : E \rightarrow E/\langle P \rangle$.
 - 33: **end if**
 - 34: $M = M \cup \{(H(\mathcal{O}), j(E/\langle P \rangle))\}$ and $m = m + 1$.
 - 35: **end if**
 - 36: **if** $m = \#\mathfrak{S}$ **then**
 - 37: Return M .
 - 38: **end if**
 - 39: Concatenate NewList and $[E/\langle P \rangle, K, \text{List}_1, \text{List}_2]$.
 - 40: **end for**
 - 41: **end for**
 - 42: $\text{List} = \text{NewList}$.
 - 43: **end for**
 - 44: **return** M .
-

Proposition 3. *Assuming Claim 1, OrdersToJInvariantBigSet is correct.*

Proof. By Claim 1 and our choice of L , we see that each maximal order types in $\mathcal{B}_{p,\infty}$ is obtained as the right order of a L -ideal. Thus, we need to prove that our algorithm goes through every possible L -ideal and that it computes correctly the j -invariant associated with the right orders of those ideals. We will make our reasoning over isogenies and the Deuring correspondence will allow us to conclude the result over ideals.

Every cyclic L -isogeny can be factored as $\varphi_n \circ \dots \circ \varphi_1$ where φ_i is an isogeny of degree L_i . When R_i, S_i is a basis of $E[L_i]$, then all cyclic subgroups of order L_i are generated by an element $[C]R_i + [D]S_i$. There is a 1-to-1 correspondence between cyclic subgroups of order L_i and cyclic isogenies of degree L_i . Since $R_{0,i}, S_{0,i}$ is a basis of $E_0[L_i]$ and $\varphi_{i-1} \circ \dots \circ \varphi_1$ has degree coprime with L_i , the two points R_i, S_i are a basis of $E[L_i]$ and so this proves that our enumeration covers all possible isogenies of degree L_i at each iteration of index i of the loop in line 19.

Thus, at the end of the loop we have covered all L -isogenies, and this means that if our ideal computation is correct, then we have covered all maximal order types and our algorithm is correct.

Remains to prove that the ideal $I_1 \cap \dots \cap I_i$ where each $I_j = \mathcal{O}_0 \langle \alpha_0(C_j + D_j\bar{\alpha}), L_j \rangle$ is the ideal corresponding to the isogeny $\varphi_i \circ \dots \circ \varphi_1$ where each φ_j has kernel $\varphi_{j-1} \circ \dots \circ \varphi_1([C_j]R_{0,j} + [D_j]S_{0,j})$ or equivalently that the kernel of $\varphi_i \circ \dots \circ \varphi_1 = \sum_{j=1}^i ([C_j]R_{0,j} + [D_j]S_{0,j})$.

For that, it suffices to prove the result for each coprime factor of the degree, so we need to prove that $E_0[I_j] = \langle [C_j]R_{0,j} + [D_j]S_{0,j} \rangle$. Let us go back to the definition of an ideal kernel given in Section 2. We have $E_0[I_j] = \{P, \beta(I_j) = 0 \forall \beta \in I_j\}$. Since I_j contains $L_j\mathcal{O}_0$, it is clear that the kernel must be a subgroup of $E_0[L_j]$. Since multiplication in $\mathcal{B}_{p,\infty}$ amounts to composition of the corresponding isogenies, it suffices to verify that $\ker \alpha_0(C_j + D_j\bar{\alpha}) \cap E_0[L_j] = \langle [C_j]R_{0,j} + [D_j]S_{0,j} \rangle$.

First, note that we have $\ker \alpha_0 \cap E_0[J_j] = \langle R_{0,j} \rangle$ by definition of α_0 and $R_{0,j}$. Then, we have $([C_j] + [D_j]\bar{\alpha})([C_j]R_{0,j} + [D_j]S_{0,j}) = ([C_j] + [D_j](\bar{\alpha})([C_j] + [D_j]\alpha))(R_{0,j}) = [\mathbf{n}(C_j + D_j\alpha)]R_{0,j}$. This proves that we have $\langle [C_j]R_{0,j} + [D_j]S_{0,j} \rangle \subset \ker \alpha_0(C_j + D_j\bar{\alpha}) \cap E_0[L_j]$.

By definition of α the scalar $\mathbf{n}(C_j + D_j\alpha)$ is coprime with L_j and so the endomorphism $C_j + D_j\alpha$ is a bijection on $E_0[L_j]$. Thus, there cannot be another subgroup than $\langle [C_j]R_{0,j} + [D_j]S_{0,j} \rangle$ that is sent to $\langle R_{0,j} \rangle$ and this concludes the proof that $\ker \alpha_0(C_j + D_j\bar{\alpha}) \cap E_0[L_j] = \langle [C_j]R_{0,j} + [D_j]S_{0,j} \rangle$.

With that last fact, we have proven the point.

Complexity analysis. Below, we give as Theorem 1, a complexity statement for Algorithm 1. We derive this result from smaller statements for all the main Steps of Algorithm 1. The proofs of these statements include a more detailed description of the steps when needed. When a step has a complexity that will end-up being negligible before the total cost, we will sometimes not bother with a precise statement.

Note that all operations involving manipulations of the list M can be done efficiently in $O(1)$ using adequate data structure so we do not analyze this part of the computation.

Since we look for an asymptotic statement, we assume that the size n of the good-degree basis used is bigger than 6 so we can apply Proposition 2.

Proposition 4. *Under GRH, Step 2 can be executed $O(\text{poly}(\log p))$.*

Proof. The first step can be performed using an algorithm that was described as part of the proof of [EHL⁺18, Proposition 3]. The idea is the following. Select the smallest fundamental discriminant d such that p is inert in the ring of integer R_d of $\mathbb{Q}(\sqrt{d})$. Note that it can be proven that $d = O(\log p^2)$ under GRH. Then, we know that there are supersingular curves that admit an embedding of R_d in their endomorphism ring. The j -invariants of these curves are the roots to the Hilbert class polynomial H_d . It suffices to find one root of H_d over \mathbb{F}_p to get a supersingular curve E_0 whose endomorphism ring will contain the Frobenius π and an endomorphism ι of norm d . This endomorphism can be found by computing all the isogenies of degree d with the Vélú formulae. Since the suborder $\langle 1, \iota, \pi, \iota \circ \pi \rangle$ has an index in $O(d) = O(\text{poly}(\log p))$ inside $\text{End}(E_0)$, and recovering the full endomorphism ring can be done in $O(\text{poly}(\log p))$.

This proves the result for Step 2.

Proposition 5. *Step 3 and Step 4 can be executed in $O(\text{poly}(\log(p)))$ and the output α_0 and α have coefficients in $O(\text{poly}(p))$ over the canonical basis of $\mathcal{B}_{p,\infty}$.*

Proof. First, note that we can fix a basis of \mathcal{O}_0 with coefficients (over the canonical basis of $\mathcal{B}_{p,\infty}$) in $O(p)$ (see [KLPT14, Section 2.3] for instance).

The Steps 3 and 4 can be solved in a similar manner despite a final goal that is quite different. For Step 3, it is sufficient to find a β_0 such that the matrix of the action of β_0 on a basis of the L -torsion has two distinct eigenvalues. Then, we can take $\alpha_0 = \beta_0 - \lambda$ where λ is one of the two eigenvalues. Note that this is equivalent to saying that β_0 needs to have two distinct eigenvalues mod L_i for all $1 \leq i \leq n$.

For Step 4, a sufficient condition to obtain α such that $\gcd(\mathbf{n}(x + y\alpha), L) = 1$ for all x, y with $\gcd(x, y, L) = 1$ is to have that the matrix of α over a basis of the L_i -torsion with no eigenvalues for all $1 \leq i \leq n$.

The existence and value of these eigenvalues mod L_i for a quaternion element β can be verified directly by computing the roots of the polynomial $X^2 + \text{tr}(\beta)X + \mathbf{n}(\beta)$ mod L_i .

Thus, to solve the two steps, we can apply the following method. First, for each L_i dividing L , find one element $\beta_{0,i}$ (resp. α_i) in $\mathcal{O}_0/L_i\mathcal{O}_0$ with two distinct (resp. no) eigenvalues mod L_i . Since the ring $\mathcal{O}_0/L_i\mathcal{O}_0$ is isomorphic to $\mathbb{M}_2(\mathbb{Z}/L_i\mathbb{Z})$ it is clear that a solution can be found by enumerating over the L_i^4 elements of $\mathcal{O}_0/L_i\mathcal{O}_0$.

Then, the final element β_0 (resp. α) can be obtained by CRT (doing this coefficient-wise over the basis of \mathcal{O}_0). The coefficients of β_0 (resp. α) will have size $O(L)$ over the basis of \mathcal{O}_0 and so we get the desired result by taking into

account the coefficients of this basis over the canonical basis of $\mathcal{B}_{p,\infty}$. For each L_i , we have to enumerate at most L_i^4 quaternion elements, so the final complexity statements follows from the complexity of computing modular squareroot and the complexity of CRT.

Proposition 6. *The FOR loop in line 5 can be executed in $O(\text{poly}(\log(p)))$.*

Proof. The loop is repeated $O(\log p)$ times. Computing a basis is a very standard task and it can be done in $O(\text{poly}(\log(p)))$ since $m_i = O(\log p)$. The generator $R_{0,i}$ can be computed in $O(\text{poly}(\log(p)))$ using the algorithm described in [GPS17] to find the kernel of an ideal. Evaluating the endomorphism α can be done by evaluating a basis of $\text{End}(E_0)$ and then performing the scalar multiplications corresponding to the coefficients of α in this basis. The first part can be done in $O(\text{poly}(\log(p)))$ with the ideas of the proof of Proposition 4 by choice of E_0 and the second can also be done in $O(\text{poly}(\log(p)))$ by the size bound given on the coefficients of α in Proposition 5.

The main computational task of `OrdersToJInvariantBigSet` is quite clearly performed during the loop in line 15. This FOR loop contains two inner loops. We will incrementally provide complexity statements for each of these loops in orders to clearly decompose the cost of each operations.

Proposition 7. *At index $i < n - 6$, Step 26 produces a polynomial over \mathbb{F}_{p^2} of degree smaller than L_i in $O(M_{\mathbb{P}}(L_i) \log(L_i))$ operations over $\mathbb{F}_{p^{m_i}}$. This polynomial defines uniquely the isogeny φ . At index $n - 6 \leq i \leq n$, $L_i = \ell_i^{e_i}$ for some prime $\ell_i \leq 13$ (the 6-th prime) and Step 26 produces e_i polynomials of degree ℓ_i over \mathbb{F}_{p^2} in $O(e_i^2)$ operations over $\mathbb{F}_{p^{m_i}}$. These polynomials uniquely defines the isogeny φ .*

Proof. It can be shown with the Vélú formulaes [Vél71], that to represent an isogeny it suffices to compute its kernel polynomial, i.e., the polynomial whose roots are the x-coordinate of the points of the kernel and that this polynomial is always defined over \mathbb{F}_{p^2} even if the kernel points are not.

From a single kernel point, the $O(L_i)$ points of the kernel can be generated in $O(L_i)$ operations over $\mathbb{F}_{p^{m_i}}$. Then, the kernel polynomial can be constructed with complexity $O(M_{\mathbb{P}}(L_i) \log(L_i))$ from its roots.

When $n - 6 \leq i \leq n$, we can write $L_i = \ell_i^{e_i}$ for some prime $\ell_i = O(1)$ and then, we can factor our isogeny of degree L_i as e_i isogenies of degree ℓ_i . All these isogenies can be computed in time $O(e_i^2)$ from a kernel generator (see [JDF11] for more on this topic).

Proposition 8. *There exists a constant C such that, at any index $i \leq n - 6$, the number of \mathbb{F}_p -operations executed in Steps 27,28 of the FOR loop in line 19 is upper bounded by $C(n - i) \log p M_{\mathbb{P}}(\log p)$. When $i > n - 6$, we have the upper-bound $C(n - i) \log(p)^2 M_{\mathbb{P}}(\log p)$*

Proof. For each $i < j \leq n$, we need to evaluate the polynomial produced by Step 26 on the points R_{i+1}, \dots, R_n and S_{i+1}, \dots, R_n . By Proposition 7, when $i \leq$

$n - 6$, each evaluation costs $O(L_i)$ operations over $\mathbb{F}_{p^{m_j}}$. The cost of arithmetic over $\mathbb{F}_{p^{m_j}}$ in operations over \mathbb{F}_p is upper-bound by $C'M_{\mathbb{P}}(m_j)$ for some constant C' . So we get the result from $L_i = O(\log p)$ and $m_j = O(\log p)$ by Lemma 2.

When $i > n - 6$, there are $e_i = O(\log(p))$ polynomials of degree $\ell_i = O(1)$ and we get the desired result.

Remark 2. The Vélu's algorithm from [BFLS20] cannot be applied here because the kernel of the isogenies and the point on which the evaluation is performed do not live in the same extension.

Proposition 9. *There exists a constant C such that, at any index $i \leq n - 6$, the number of binary operations executed in each execution of the FOR loop in line 19 is upper bounded by $C\Phi(L_i)(n - i) \log p \log(p) M_{\mathbb{P}}(\log p) M_{\mathbb{Z}}(\log p)$. When $n - 6 < i < n$, the number of binary operations is smaller than*

$$C\Phi(L_i) \log(p)^2 M_{\mathbb{P}}(\log p) M_{\mathbb{Z}}(\log p).$$

Proof. For each execution of this loop, the number of iteration is exactly $\Phi(L_i)$. Arithmetic over quaternion orders and ideals such as intersection and right order computation can be performed in $O(M_{\mathbb{Z}}(\log p))$ (because the coefficients have size in $O(\log p)$ and these operations are simple linear algebra in dimension 4). Then, the hash can be computed in $O(\log p^{1+\varepsilon})$. Thus, the total cost of the loop is $O(\Phi(L_i) \log p^{1+\varepsilon})$ for quaternion operations for any i . This is negligible compared to other operations when $i < n$.

The verification that $H(\mathcal{O}) \in \mathfrak{S}$ and insertion in the hash map operation can be done in $O(1)$ with the appropriate hash-map structure.

Then, there are the cost of operations over \mathbb{F}_p for the isogeny computation. To derive the total complexity, we apply Propositions 7 and 8. Arithmetic over \mathbb{F}_p takes $O(M_{\mathbb{Z}}(\log p))$ binary operations. For any i , the kernel computation is negligible. When $i \leq n - 6$, the isogeny computation takes

$$O(\Phi(L_i) \log p \log(p) M_{\mathbb{P}}(\log p) M_{\mathbb{Z}}(\log p)),$$

then the evaluations take $O((n - i)\Phi(L_i) \log p M_{\mathbb{P}}(\log p) M_{\mathbb{Z}}(\log p))$.

Similarly, when $n - 6 < i < n$, this cost is replaced by

$$O(\Phi(L_i) \log(p)^2 M_{\mathbb{P}}(\log p) M_{\mathbb{Z}}(\log p)).$$

Proposition 10. *There exists a constant C such that, at any index $i \leq n - 6$, the number of binary operations executed in the FOR loop in line 17 is upper bounded by $C \prod_{j=1}^i \Phi(L_j)(n - i) \log p^{3+\varepsilon}$. When $n - 6 < i < n$, it is upper bounded by $C \prod_{j=1}^i \Phi(L_j)(n - i) \log p^{2+\varepsilon}$. When $i = n$, it is upper-bound by $C(\#\mathfrak{S} \log p^{2+\varepsilon} + p \log p^{1+\varepsilon})$.*

Proof. There are $\Phi(L_i)$ cyclic subgroups of order L_i . Thus at index $i < n$, the size of **List** is $\prod_{j=1}^{i-1} \Phi(L_j)$ and the result follows directly from Proposition 9.

When $i = n$, we perform the quaternion computations (intersection, right order and computation of the hash value) for all $\prod_{j=1}^n \Phi(L_i)$ subgroups. Thus,

since we have $\prod_{j=1}^n \Phi(L_j) = O(p)$ by Lemma 1 and Claim 1, and the cost of quaternion operations is $O(\log p^{1+\varepsilon})$ as in the proof of Proposition 9, we get that the cost for quaternion operations is $O(p \log p^{1+\varepsilon})$. The isogeny computation is only performed when the right maximal order is contained in \mathfrak{S} , thus, we can upper-bound the number of times where an isogeny is computed by $\#\mathfrak{S}$. As for all the $i \geq n - 6$, the cost of each L_n -isogeny computation is $O((\log(p))^2 M_{\mathbb{F}}(\log p) M_{\mathbb{Z}}(\log p))$ and this proves the result.

Proposition 11. *The loop in line 15 can be executed in $O(\#\mathfrak{S} \log p^{2+\varepsilon} + p \log p^{1+\varepsilon})$ binary operations.*

Proof. The total cost of the loop is directly obtained by summing over all i the bounds in Proposition 10. We start by summing over all $i \leq n - 6$. We get an upper-bound on the number of binary operation by $C \sum_{i=1}^{n-6} (n - i) \prod_{j=1}^i \Phi(L_j) \log p^{3+\varepsilon}$. So we get $O(p \log p^{1+\varepsilon})$ after applying Proposition 2.

For $n - 6 < i < n$, we get the upper-bound

$$C \sum_{i=n-5}^{n-1} (n - i) \prod_{j=1}^i \Phi(L_j) \log p^{2+\varepsilon}.$$

Since by Lemma 1, $\Phi(L_n) > C_0/2 \log p$, we have $\prod_{j=1}^i \Phi(L_j) \leq C' p / \log p$ for some constant C' and any $i < n$. Thus, since there is a constant number of summands, we get the cost is in $O(p \log p^{1+\varepsilon})$ for those indices.

Finally, at $i = n$, we can apply directly the bound from Proposition 10. The final cost is $O(\#\mathfrak{S} \log p^{2+\varepsilon} + p \log p^{1+\varepsilon})$.

All the results above lead directly to the following theorem.

Theorem 1. *Under Claim 1, on input p and \mathfrak{S} , `OrdersToJInvariantBigSet` can be executed in*

$$O(\#\mathfrak{S} \log p^{2+\varepsilon} + p \log p^{1+\varepsilon})$$

binary operations. The space requirement of `OrdersToJInvariantBigSet` is in $O(p \log p)$.

Proof. The cost of Step 1 is negligible before all the other computations. The result follows from all the propositions proven in this section.

For the space requirement, the proof is quite similar. After each iteration of the loop for an index $i \leq n$, the list contains $\prod_{j=1}^i \Phi(L_i)$ entries of the form $E, K, P_{i+1}, \dots, P_n, Q_{i+1}, \dots, Q_n$ where E is an elliptic curve, K is an ideal of norm $\prod_{j=1}^i L_i$ and P_j, Q_j are points of order L_i . Thus, this can be represented in $O(\prod_{j=1}^i \Phi(L_i) (\log p + (n - i) \log(p^2)))$. Using Proposition 2 for $i \leq n - 6$ we see that this takes $O(p \log p)$ space. With $\Phi(L_n) = \Theta(\log(p))$, we get that $\prod_{j=1}^i \Phi(L_i) = O(p / \log p)$ and so the memory required for indices $n - 6 \leq i < n$ is also in $O(p \log p)$. Finally, at index $i = n$ the second term is zero and so we get $O(p \log p)$.

Finally, the space required by the list M is $O(\#\mathfrak{S} \log p) = O(p \log p)$, and this concludes the proof.

Good choice of primes. The analysis we provided above for both `OrdersTojInvariantSmallSet` and `OrdersTojInvariantBigSet` does not assume anything on the prime p . There are some "nice" choices of primes p for which we could basically gain a factor $\log p$ over all elliptic curve operations by having all the required torsion point defined over \mathbb{F}_{p^2} (thus saving the cost of operations over big \mathbb{F}_p -extensions). Since we are interested in a generic statement, we do not bother with these marginal gains.

In the context of applying `OrdersTojInvariantBigSet` to the CRT method, this idea will have its importance in the concrete choice of primes p_i . However, due to the linear dependency in p , it does not appear possible to select all CRT primes among these "nice" primes. And so the CRT complexity will depend on the worst-case complexity of our algorithm `OrdersTojInvariantBigSet`.

4 Computation of the Hilbert class polynomial

In this section, we introduce our two algorithms to compute Hilbert class polynomial. In Section 4.1, we present the direct algorithm, and in Section 4.2, we introduce the CRT algorithm and we also provide a comparison with Sutherland's CRT algorithm from [Sut11].

4.1 Computing the class polynomial modulo a non-split prime.

Let us fix some negative discriminant D and a non-split prime p in $\mathbb{Q}(\sqrt{D})$ (and we assume further that p is coprime to the conductor of \mathfrak{D}). We write h for the class number of \mathfrak{D} . For simplicity, we assume below that the factorization of D is known so this step is not part of our estimates.

Our goal in this section is to explain how to compute the Hilbert class polynomial $H_D(X) \pmod{p}$. This polynomial can be reconstructed from its roots that are j -invariants of some supersingular curves over \mathbb{F}_{p^2} .

When p is split, the roots correspond to ordinary curves. In Sutherland's algorithm they are obtained in two main steps algorithm [Sut11]: start by identifying one root, and then enumerate through all the roots using the action of the class group $\text{Cl}(\mathfrak{D})$ through isogenies of small degree.

When p is non-split, we have three main steps. We start to do something very similar to what is done for ordinary curves, but instead of working directly with the roots, we first use the Deuring correspondence, and identify the roots from the the corresponding maximal orders of the quaternion algebra $\mathcal{B}_{p,\infty}$. The class group action of $\text{Cl}(\mathfrak{D})$ can be realized much more efficiently in that case because all operations in $\mathcal{B}_{p,\infty}$ are essentially linear algebra.

The Deuring correspondence states that the set of maximal order we obtain in this manner are isomorphic to the endomorphism rings of the elliptic curves we want to compute. Thus, we constitute a set $\mathfrak{S}_D(p)$ of maximal order type and we can apply `OrdersTojInvariantBigSet` on this set to compute the j -invariants we need. This execution constitutes our third step.

The fourth and final step in our algorithm is common with the third step of the ordinary case: recover the polynomial H_D from its roots. Note that this is done using standard polynomial arithmetic.

This is described more precisely in Algorithm 2.

Algorithm 2 SupersingularHilbertComputation(D, p)

Input: A discriminant $D < 0$ and a prime p such that p is non-split in, and coprime to the conductor of, \mathfrak{D} the quadratic order of discriminant D .

Output: $H_D(X) \pmod p$.

- 1: Find a maximal order \mathcal{O} in $\mathcal{B}_{p, \infty}$ with $\mathfrak{D} \hookrightarrow \mathcal{O}$.
 - 2: Use the action of $\text{Cl}(D)$ to find $\mathfrak{S}_D(p)$, the set of types of maximal orders in $\mathcal{B}_{p, \infty}$ with an optimal embedding of \mathfrak{D} . To each type $\mathcal{O} \in \mathfrak{S}_D(p)$ is associated a number $n_{\mathcal{O}} \geq 1$ counting the multiplicity of the type \mathcal{O} in this enumeration. The set of these multiplicities is write $N_D(p)$.
 - 3: Compute $J_D(p) = \text{OrdersToJInvariantSmallSet}(p, \mathfrak{S}_D(p))$ (or use `OrdersToJInvariantBigSet` if this is more efficient).
 - 4: Reconstruct $H(X) = \prod_{j_D(p) \in J} (X - j) \pmod p$ from $\mathfrak{S}_D(p)$, $J_D(p)$ and $N_D(p)$.
 - 5: **return** Return $H(X)$.
-

Proposition 12. *SupersingularHilbertComputation is correct and it can be executed in*

$$O(\sqrt{|D|}(\log(|D|)^{2+\varepsilon} \log(p)^{1+\varepsilon} + \log(|D|)^\varepsilon \log(p)^{2+\varepsilon}) + p \log(p)^{1+\varepsilon}),$$

binary operations and requires $O((p+h(D)) \log p)$ memory when `OrdersToJInvariantBigSet` is used in Step 1 (assuming GRH and Claim 1), and

$$O(\sqrt{|D|}(\log(|D|)^\varepsilon \log(p)^{5+\varepsilon} + \log(|D|)^{2+\varepsilon} \log(p)^{1+\varepsilon})),$$

binary operations and requires $O(h(D) \log(p))$ space with `OrdersToJInvariantSmallSet` (under GRH and the heuristics from [KLPT14]).

Proof. Correctness follows from [Onu21, Theorem 3.4]. We will now explain how each step can be performed and what is the best known complexity.

Step 1. This task has already been solved in the context of generating backdoor curves to the SIDH scheme [QKL⁺21] and generating keys for the Seta encryption scheme [DFFdSG⁺21]. First, we need to solve a quadratic equation over \mathbb{Q} to find $a, b, c, d \in \mathbb{Q}$ such that $\mathbb{Z}[a+ib+jc+kd]$ is the quadratic order of discriminant D . This can be done using Simon's algorithm [Sim05] to solve quadratic forms in dimension 4. The complexity of Simon's algorithm is polynomial in the logarithm of the determinant once the factorization of the determinant is known and the size of the output is also logarithmic in the determinant. In our case, the quadratic form we consider is basically $b, c, d, e \mapsto (qb^2 + p(c^2 + qd^2) - e^2D)$ and its determinant is equal to $p^2q^2D2^f$ for some small integer f . The computation of the full factorization is sub-exponential in $\log(pD)$.

Now that $\theta = a + ib + jc + kd$ has been computed, we need to find a maximal order \mathcal{O} containing it. Let us take A as the smallest common denominator of a, b, c, d , we have $A = O(\text{poly}(\log(pD)))$. Then $A\theta \in \mathcal{O}_0$ where \mathcal{O}_0 is any maximal order containing the sub-order $\langle 1, i, j, k \rangle$. Since $A\theta \in \mathcal{O}_0$, the right order of the ideal $I = \mathcal{O}_0 A\theta + \mathcal{O}_0 C$ contains θ . We can set $\mathcal{O} = \mathcal{O}_R(I)$ and \mathcal{O} can be computed in $O(\text{poly}(\log(p|D|)))$. Hence, this step can be performed in sub-exponential in $\log p|D|$ and is negligible compared to the rest of the computation.

Step 2. We go from one maximal order type to all maximal order types of interest by using the group action of the class group in a manner similar to what is used by Sutherland in [Sut11]. For that, we use a polycyclic representation of $\text{Cl}(D)$ as introduced in [Sut11, Section 5]. But, in our case, instead of isogeny computation, we can simply use arithmetic over quaternions through the action of ideals of the form $\mathcal{O}(\theta - \lambda) + \mathcal{O}\ell$ on the set of maximal orders containing θ which cover all maximal order types we need. Any group action computation for an ideal of norm ℓ takes $O(\log(\ell))$. Thus, using the same estimates than in [Sut11, Lemma 7] where we see that under GRH the biggest norm of an ideal involved a polycyclic representation is in $O(\log^2(D))$, we see that this part can be performed in $O(h \log(|D|^\varepsilon)) = O(\sqrt{|D|} \log(|D|^\varepsilon))$.

We can hash (with the function introduced in Section 3.1) all the maximal order types obtained in this manner to create the set $\mathfrak{S}_D(p)$ in $O(\sqrt{|D|} \log(|D|^\varepsilon) \log(p)^{1+\varepsilon})$.

Step 3. This step consists simply in the execution of `OrdersTojInvariantBigSet` on the set $\mathfrak{S}_D(p)$ computed in Step 2. Thus, by Theorem 1 and the estimates on h , the complexity of this step is $O(\sqrt{|D|} \log(|D|^\varepsilon) \log(p)^{2+\varepsilon} + p \log(p)^{1+\varepsilon})$. Alternatively, it is possible to use the `OrdersTojInvariantSmallSet` algorithm and obtain a complexity of $O(\sqrt{|D|} \log(|D|^\varepsilon) \log(p)^{5+\varepsilon})$.

The reconstruction step. The complexity of this step is $O(\sqrt{|D|} \log(|D|)^{2+\varepsilon} \log(p)^{1+\varepsilon})$ as was proven in [Sut11].

The total complexity. Putting together all the results above, we get the complexity result.

Space complexity. In terms of memory requirement, the polynomial reconstruction requires $O(h(D) \log p)$ space (since we only need to store two levels of the product tree at the same time). The set of maximal orders takes $O(h(D) \log p)$ space. For the variant with `OrdersTojInvariantBigSet`, the results follows from Theorem 1. When `OrdersTojInvariantSmallSet` is used, the computation of each j -invariant is done sequentially, and so the space requirement is optimal and is $O(h(D) \log p)$. \square

Which version is optimal? We obtain two versions of `SupersingularHilbertComputation` depending on whether we use `OrdersTojInvariantSmallSet` or `OrdersTojInvariantBigSet`. For a given value of D , the former has a better asymptotic complexity. However, for primes p in $O(\sqrt{|D|} \log(|D|)^{4+\varepsilon})$, the latter has a better complexity.

For a range of medium-sized primes, this algorithm will have the best known complexity. The cut-off with the CRT method (whose complexity is $O(|D|^{1+\varepsilon})$) (as we will see below) happens for $p = O(2^{|D|^{1/10}})$.

4.2 Application to the CRT method and comparison with existing method.

In this section, we present and analyze the algorithm yielded by applying the CRT method from [Sut11] on top of the `SupersingularHilbertComputation` algorithm introduced in the previous section. The algorithm is pretty straightforward, and the only constraint on the selection of the small primes.

Algorithm 3 `HilbertComputation`(D, p)

Input: A discriminant $D < 0$ and a prime p .

Output: $H_D(X) \pmod p$.

- 1: Set B_D as the upper-bound on the coefficients of $H_D(X)$ over \mathbb{Z} .
 - 2: Set $\mathcal{P}_\mathfrak{D} = \{\}$, $P = 1$, $q = 3$.
 - 3: **while** $P < B_D$ **do**
 - 4: **if** q is prime and q is non-split in and coprime to the conductor of \mathfrak{D} **then**
 - 5: $P \leftarrow q \cdot P$, $\mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \cup \{q\}$.
 - 6: **end if**
 - 7: $q \leftarrow q + 2$
 - 8: **end while**
 - 9: Perform the precomputations for the explicit CRT mod p using $\mathcal{P}_\mathfrak{D}$.
 - 10: **for** $q \in \mathcal{P}_\mathfrak{D}$ **do**
 - 11: $H_{\mathfrak{D},q}(X) \leftarrow \text{SupersingularHilbertComputation}(D, q)$.
 - 12: Update the CRT sums for each coefficient of $H_{\mathfrak{D},q}(X)$.
 - 13: **end for**
 - 14: Perform the postcomputation for the explicit CRT to obtain $H_D(X) \in \mathbb{F}_p[X]$.
 - 15: **return** Return $H_D(X)$.
-

Proposition 13. *Under GRH and Claim 1, the complexity of `HilbertComputation` is $O(|D| \log^{3+\varepsilon} |D|)$. The space requirement is $O(\sqrt{|D|}(\log^{2+\varepsilon} |D| + \log p))$*

Proof. Under GRH, we can lower bound the probability that a given prime is non-split in \mathfrak{D} by $1/2$. Then, following the same reasoning as in [Sut11], it can be shown that we have $\#\mathcal{P}_\mathfrak{D} = O(\log B_D / \log(B_D))$ and $\max_{q \in \mathcal{P}_\mathfrak{D}} q = O(\log(B_D))$. With the usual $\log B_D = O(\sqrt{|D|} \log^{1+\varepsilon} |D|)$ that holds under GRH, we get that we can take $O(\sqrt{|D|})$ primes with $\max_{q \in \mathcal{P}_\mathfrak{D}} q = O(\sqrt{|D|} \log(|D|))$. By Proposition 12, the dominant complexity of each CRT computation is the cost of the polynomial reconstruction. Thus, following the complexity estimates from [Sut11], we obtain that the overall complexity estimate is $O(|D| \log(|D|)^{3+\varepsilon})$. The space complexity follows from the space requirements computed in [Sut11] and Proposition 12.

Practical comparison with the CRT algorithm based on ordinary curves from [Sut11]. We can see that the complexity reported in Proposition 15 is the same as Sutherland’s algorithm from [Sut11]. Even if the two algorithms have the same asymptotic complexity, they might not have the same practical efficiency. Below, we try to see which one could be faster in practice.

First, note that the set of CRT primes are very different between the two methods. Since the probability that a given prime can be selected in \mathcal{P}_D in our algorithm `HilbertComputation` is $1/2$, the primes that we select are going to be much smaller than the ones used in Sutherland’s algorithm.

Now, we look at the concrete algorithmic steps. Let us start with the polynomial reconstruction step as it is the asymptotically dominant step. We remind that the concrete complexity of this step is $O(\sqrt{|D|} \log(|D|)^{2+\epsilon} \log(p))$. It is pretty similar in both algorithms and we argue that the practical cost should be roughly the same. This is not completely obvious since the primes will not have the same size in the two cases and the roots are defined over \mathbb{F}_p for ordinary curves against \mathbb{F}_{p^2} over supersingular curves. First, the size of the primes does not really matter because the product $\prod_{i=1}^n p_i$ have roughly the same size in the two cases and the complexity of the reconstruction is linear in $\log(p_i)$ for all i . Second, since in the supersingular case, the Galois conjugate (by the action of the Frobenius) of a root of $H_D \pmod{p_i}$ is also a root, by building the remainder tree from polynomials of the form $(X - j)(X - j^{p_i}) \in \mathbb{F}_{p_i}[X]$, we see that we can make the entire computation over \mathbb{F}_{p_i} as in the ordinary case (and thus avoid the constant overhead brought by multiplications over $\mathbb{F}_{p_i^2}$). We conclude from this brief reasoning that the reconstruction cost will be essentially the same in the two cases.

Now, if we forget the reconstruction step, we see that using supersingular curve offers an asymptotic advantages. Indeed, in Steps 1, 2 and 3 of our algorithm, the dominant step is the execution of `OrdersTojInvariantBigSet` in Step 3, which has a $O(|D| \log(|D|)^{2+\epsilon})$ complexity (if we consider the executions over all primes $p_i \in \mathcal{P}_D$ and we use $\log(p_i) = O(\log(|D|))$). In particular, this is smaller than the $O(|D| \log(|D|)^{5/2+\epsilon})$ that dominates that part of the computation in Sutherland’s algorithm (corresponding to the computation of one curve with the correct endomorphism ring).

This is the first reason that suggests that the supersingular case might be more efficient than the ordinary one, but this is not the main one. The main reason behind the practical speed-up we hope to obtain is that we can use smaller primes. Indeed, the expected maximum of our primes is in $O(\sqrt{|D|} \log(|D|))$ (against $O(|D| \log(|D|)^{1+\epsilon})$ for ordinary curves). Moreover, we can take all the small primes that satisfy the redosity condition. In particular, we will be able to use a good portion of primes significantly smaller than $\sqrt{|D|}$.

We hope that the very small primes will give a nice improvement in practice because for these primes, some of the roots will have big multiplicities, which should help perform every steps more efficiently in practice (for example there will be less than $O(\sqrt{|D|})$ j -invariants to compute in that case) and it should improve the practical efficiency.

Note that there is also a good potential for practical improvement by carefully selecting the primes in \mathcal{P}_D and choosing the good-degree basis used for each of those primes in order to minimize the degree of the extension required to compute the isogenies in `OrdersTojInvariantBigSet`. A selection is also performed in the algorithm of Sutherland to help improve the cost of finding one curve with the good cardinal, so it would be natural to do the same thing in our case.

Even if `OrdersTojInvariantBigSet` proves to be too slow to beat the version of Sutherland by using only supersingular primes, it is clear that it is worth considering an hybrid set of primes \mathcal{P}_D containing a mix of supersingular and ordinary primes to obtain the best efficiency as the computation will be definitely very fast for a lot of small non-split primes.

Further improvement: batching class polynomial computation. `OrdersTojInvariantBigSet` can be easily modified to handle several sets $\mathfrak{S}_1, \dots, \mathfrak{S}_k$ more efficiently than k executions of `OrdersTojInvariantBigSet` for each \mathfrak{S}_i .

Thus, if we have several discriminants D_1, \dots, D_k , and a prime p in $\bigcap_{1 \leq i \leq k} \mathcal{P}_{D_i}$. A good part of the computations performed to compute $H_{D_1}, \dots, H_{D_k} \pmod p$ can be done at the same time at a reduced cost.

Moreover, if some H_{D_i} have some common roots, the common divisors could be constructed once and for all.

Thus, our new method could be used to batch efficiently the computation of several class polynomial at the same time.

5 Computation of the modular polynomials

In this section, we introduce our new algorithms to compute modular polynomials. In Section 5.1, we present the direct algorithm, and in Section 5.2, we describe the CRT algorithm and compare the various existing algorithms of the literature.

5.1 Direct modular polynomial computation with supersingular curves

Let us take a prime level ℓ and a prime p . To make a direct computation of $\Phi_\ell(X, Y) \pmod p$ it suffices to identify $\ell + 1$ distinct j -invariants and for each of those, to interpolate $\Phi_\ell(j, Y)$ from its roots that are the $\ell + 1$ ℓ -isogenous j -invariants.

The idea of our algorithm for modular polynomials follows the same principle as the class polynomials algorithm. There is a slight difference because modular polynomials are bivariate but it does not change the generic principle of the algorithm. Indeed the full polynomial $\Phi_\ell \pmod p$ is interpolated from the $\Phi_\ell(j_i, Y)$ for $\ell + 1$ j -invariants $(j_i)_{0 \leq i \leq \ell}$ of elliptic curves defined in $\overline{\mathbb{F}_p}$. Each univariate polynomial $\Phi_\ell(j_i, Y)$ is reconstructed from its roots, that are the j -invariants of curves ℓ -isogenous to j_i .

In `SupersingularModularComputation`, we compute the maximal orders that correspond to all the j -invariants that we need under the Deuring correspondence, and then, we apply `OrdersTojInvariantBigSet` (or `OrdersTojInvariantSmallSet`) to find the required j -invariants.

Algorithm 4 `SupersingularModularComputation`(p, ℓ)

Input: A prime p and a prime ℓ such that $\lceil p/12 \rceil + 1 > \ell$.

Output: $\Phi_\ell(X, Y) \pmod p$.

- 1: Compute a set $\mathfrak{S}_0 = \{\mathcal{O}_1, \dots, \mathcal{O}_{\ell+1}\}$ of distinct maximal order types in $\mathcal{B}_{p, \infty}$.
 - 2: Set $\mathfrak{S}_1, \dots, \mathfrak{S}_{\ell+1} = \{\}, \dots, \{\}$.
 - 3: **for** $i = 1$ to $\ell + 1$ **do**
 - 4: Compute $I_{i,1}, \dots, I_{i,\ell+1}$, the $\ell + 1$ left \mathcal{O}_i -ideals of norm ℓ .
 - 5: $\mathfrak{S}_i \leftarrow \{\mathcal{O}_R(I_{i,k}) \mid 1 \leq k \leq \ell + 1\}$.
 - 6: **end for**
 - 7: $\mathfrak{S} = \bigcup_{0 \leq i \leq \ell+1} \mathfrak{S}_i$.
 - 8: Compute $J = \text{OrdersTojInvariantSmallSet}(p, \mathfrak{S})$ (or use `OrdersTojInvariantBigSet`(p, \mathfrak{S}) if this is more efficient).
 - 9: Divide J as $J_0, J_1, \dots, J_{\ell+1}$ corresponding to $\mathfrak{S}_0, \mathfrak{S}_1, \dots, \mathfrak{S}_{\ell+1}$.
 - 10: **for** $i = 1$ to $\ell + 1$ **do**
 - 11: $P(j_i, Y) \leftarrow \prod_{j \in J_i} (Y - j)$.
 - 12: **end for**
 - 13: Reconstruct $P(X, Y) \pmod p$ from the $P(j, Y)$ for $j \in J_0$.
 - 14: **return** Return $P(X, Y)$.
-

Proposition 14. `SupersingularModularComputation` can be executed in

$$O(\ell^2(\log \ell^{2+\varepsilon} \log p^{1+\varepsilon} + \log p^{2+\varepsilon}) + p \log p^{1+\varepsilon})$$

binary operations and requires $O((\ell^2 + p) \log p)$ when `OrdersTojInvariantBigSet` is used in Step 8 (assuming GRH and Claim 1), and

$$O(\ell^2(\log p^{5+\varepsilon} + \log \ell^{2+\varepsilon} \log p^{1+\varepsilon}))$$

binary operations and requires $O(\ell^2 \log p)$ space with `OrdersTojInvariantSmallSet` (under GRH and the heuristics from [KLPT14]).

Proof. With the condition on the respective size of ℓ and p , we know there are enough supersingular j -invariants defined over \mathbb{F}_{p^2} . We now briefly explain how each of the steps can be performed and what are their complexities.

By Claim 1, there is a value $L = O(p)$ such that every supersingular j -invariant are L -isogenous to a given j -invariant j_0 . Thus, under the Deuring correspondence, the $\ell + 1$ maximal order types can be computed in $O(\ell \log(p)^{1+\varepsilon})$ by starting from a canonical maximal order type $\mathcal{O}_0 \subset \mathcal{B}_{p, \infty}$ (for which there exists formulas in [KLPT14] for any p). Thus, the complexity of Step 1 is $O(\ell \log(p)^{1+\varepsilon})$.

The computation of an ℓ -ideal and of its right order can be done in $O(\log(\ell p))$. Thus, the set of maximal order types \mathfrak{S} can be computed in $O(\ell^2(\log p + \log \ell))$.

As show in [BLS12], the interpolation of all the $\Phi_\ell(j_i, Y)$, and the final interpolation of $\Phi_\ell(X, Y)$ take $O(\ell^2 \log \ell^{2+\varepsilon} \log p^{1+\varepsilon})$.

The final complexity statement follows from the results shown previously on the complexities of `OrdersTojInvariantBigSet` and `OrdersTojInvariantSmallSet`.

The space complexity of the polynomial reconstruction is $O(\ell^2 \log p)$ and the results follow from the space complexites of `OrdersTojInvariantBigSet` and `OrdersTojInvariantSmallSet`.

5.1.1 Which version is better ? Thus, we see that the first algorithm based on `OrdersTojInvariantBigSet` will be better for small values of p . We can estimate a cut-off for a value of p in $O(\ell^{2+\varepsilon})$. In that range of primes, our algorithm with `OrdersTojInvariantBigSet` has the best known generic complexity to compute $\Phi_\ell \bmod p$. For primes bigger than that, it is better to use the variant with `OrdersTojInvariantSmallSet` to avoid the quasi-linear dependency in p . For a range of medium-sized primes, this algorithm will have the best known complexity.

Space complexity. In terms of memory requirement our two algorithms are optimal and require $O(\ell^2 \log p)$.

5.2 CRT methods to compute modular polynomials from supersingular curves.

In this section, we analyse the benefit of using supersingular curves in the CRT method and compare it with the algorithm described by Bröker, Lauter and Sutherland in [BLS12]. We refer the reader to the algorithm outlined in Section 1.1. Since the CRT is typically based on a lot of very small primes, we use the variant with `OrdersTojInvariantBigSet`.

Proposition 15. *The complexity of `ModularComputation` is $O(\ell^3 \log \ell^{3+\varepsilon})$. The space requirement is $O(\ell^2 \log p)$.*

Proof. Following the same reasoning as in [Sut11], it can be shown that we have $\#\mathcal{P}_\ell = O(B_\ell / \log(B_\ell))$ and $\max_{q \in \mathcal{P}_\ell} q = O(\log(B_\ell))$. The choice of B_ℓ for the bound on the coefficients of Φ_ℓ holds under GRH. We get that we can take $O(\ell)$ primes with $\max_{q \in \mathcal{P}_\ell} q = O(\ell \log(\ell))$.

By Proposition 14, for the complexity of `SupersingularModularComputation` with `OrdersTojInvariantBigSet`, and the complexity of the CRT steps described in [BLS12], we see that the complexity of each CRT computation takes $O(\ell^2 \log(\ell)^{3+\varepsilon})$. Thus, we obtain that the overall complexity estimate is $O(\ell^3 \log(\ell)^{3+\varepsilon})$. The space complexity follows in a similar manner.

On the cost of elliptic curve operations. The complexity of our algorithm is dominated by the polynomial reconstruction step. Note that this is also the case for the BLS algorithm.

Algorithm 5 ModularComputation(p, ℓ)

Input: A prime p , $j \in \mathbb{F}_p$, a prime ℓ .

Output: $\Phi_\ell(X, Y) \pmod{p}$.

- 1: Let \bar{j} be the integer in $[0, p - 1]$ equal to $j \pmod{p}$.
 - 2: Set $B_\ell = 2^{6\ell \log \ell + 18\ell + \log(\ell + 2)}$.
 - 3: Set $\mathcal{P}_\ell = \{\}$, $P = 1$, $q = 12\ell$.
 - 4: **while** $P < B_\ell$ **do**
 - 5: **if** q is prime **then**
 - 6: $P \leftarrow q \cdot P$, $\mathcal{P}_\ell \leftarrow \mathcal{P}_\ell \cup \{q\}$.
 - 7: **end if**
 - 8: $q \leftarrow q + 2$
 - 9: **end while**
 - 10: Perform the precomputations for the explicit CRT mod p using \mathcal{P}_ℓ .
 - 11: **for** $q \in \mathcal{P}_\ell$ **do**
 - 12: $P_q(X, Y) \leftarrow \text{SupersingularModularComputation}(q, \ell)$.
 - 13: Update the CRT sums for each coefficient of $P_q(X, Y)$.
 - 14: **end for**
 - 15: Perform the postcomputation for the explicit CRT to obtain $P(X, Y) \in \mathbb{F}_p[X, X]$.
 - 16: **return** Return $P(X, Y)$.
-

However, unlike BLS, the rest of our computation is much more efficient. Indeed, if we look at the cost of the elliptic curve operation (the cost of all the elliptic curve operations inside the calls to `SupersingularModularComputation`), we see that the global complexity is only quadratic in ℓ . Indeed, we use primes q of size $O(\ell^{1+\epsilon})$ for which there exists only $O(\ell^{1+\epsilon})$ supersingular j -invariants. This means that we can simply compute them all with `OrdersTojInvariantBigSet` in quasi-linear time in ℓ , thus yielding a quadratic complexity over all CRT primes. This is why we expect our algorithm to produce a practical speed-up over the BLS method.

Batching the computation. Similarly to the Hilbert polynomial case, the set of small primes can be reused in the computations over various ℓ . In fact, the situation is even better in the modular case, because, apart from size, there are no restrictions on the primes. Thus, if we want to compute Φ_{ℓ_i} for primes $\ell_1, \ell_2, \dots, \ell_k$ of the same size, we will be able to use the same exact set \mathcal{P}_D for all the computations. Thus, only the polynomial reconstruction phase will be specific to each ℓ_i , and the rest needs to be done only once.

5.3 Comparison between different existing methods

Here, we compare the complexity of various known methods to compute modular polynomials. Note that some of these methods are proven under some quite ad-hoc heuristics (ours in particular) while some others are proven rigorously. However, all the heuristics used in the various method are plausible and have been verified experimentally. Thus, we don't expect that these heuristics should be problematic for a practical comparison.

Here is the list of algorithms to consider:

1. Our direct algorithm `SupersingularModularComputation`.
2. The direct algorithm from Robert [Rob22].
3. The CRT algorithm from BLS [BLS12].
4. Our CRT algorithm `ModularComputation`.
5. The CRT algorithm from Robert [Rob22].

The two direct algorithms are the only known algorithms for generic primes whose complexity is only quadratic in ℓ . We have stated the complexity of the two variants of `SupersingularModularComputation` in Proposition 14. The complexity of Robert’s direct algorithm is $O(\ell^2 \log p \log^{2+3u} \ell)$ for some parameter u that might be 2 or 4 (it’s not really clear which one).

On the other hand, the complexity of the three CRT algorithms is the same: it is cubic in ℓ and independant of p . BLS and ours are optimal in memory, while Robert’s is not.

We will compare these various complexities by makin the value of p vary while ℓ remains fixed.

For small values of p , `SupersingularModularComputation` with `OrdersTojInvariantBigSet` will be the best algorithm as noted in the end of Section 5.1. However, due to the linear complexity of p , this algorithm will quickly be outperformed by the other algorithms as p grows. As soon as $\ell = o(\sqrt{p})$, it will be better to use `SupersingularModularComputation` with `OrdersTojInvariantSmallSet`. The $\log \ell$ factor is only quadratic in the complexity of `SupersingularModularComputation` so it will remain better than Robert’s direct algorithm when p remains not too big. However, since the $\log p$ factor has an exponent of 5 in `SupersingularModularComputation` while Robert’s algorithm is linear in $\log p$, the latter will eventually outperform our direct algorithm. The exact cross-point will depend on the value of u .

Finally, due to the fact that the CRT methods have a running time independant of p , they will end up being more efficient than the direct algorithms when $\ell = o(\log p)$.

The practical efficiency of Robert’s algorithm is hard to estimate, but it might not be competitive with the other two due to its much bigger memory requirement (which ends being quite problematic as modular polynomial are huge). We have already argued that we expect our method to outperform the method from BLS in practice.

6 Conclusion

We have introduced several new algorithms to compute modular polynomials of level ℓ and Hilbert polynomials of discriminant D modulo a generic prime number p from supersingular curves. The direct version of our algorithmsn have complexity in ℓ^2 and $\sqrt{|D|}$ respectively for generic primes. Depending on the relative size of $\ell, |D|$ and p , we exhibit improvements over the best known asymptotic complexities for a significant range of primes.

Moreover, when applied to the CRT method, we obtain an algorithm whose complexity is the same as previously known CRT method, but with the potential to give a practical improvement (in particular in the case of modular polynomials).

It remains to see how efficient our new algorithms are in practice. There are several practical challenges to overcome before providing an implementation of the proposed algorithms (in particular related to the field extensions involved in the computations of some isogenies), and this is why we leave the concrete implementation to future work.

References

- ACNL⁺21. Sarah Arpin, Catalina Camacho-Navarro, Kristin Lauter, Joelle Lim, Kristina Nelson, Travis Scholl, and Jana Sotáková. Adventures in supersingularland. *Experimental Mathematics*, pages 1–28, 2021.
- AM93. A Oliver L Atkin and François Morain. Elliptic curves and primality proving. *Mathematics of computation*, 61(203):29–68, 1993.
- BBEL08. Juliana Belding, Reinier Bröker, Andreas Enge, and Kristin Lauter. Computing hilbert class polynomials. In *International Algorithmic Number Theory Symposium*, pages 282–295. Springer, 2008.
- BFLS20. Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *ANTS*, 2020.
- BLS12. Reinier Bröker, Kristin Lauter, and Andrew Sutherland. Modular polynomials via isogeny volcanoes. *Mathematics of Computation*, 81(278):1201–1231, 2012.
- BS07. Reinier Bröker and Peter Stevenhagen. Efficient cm-constructions of elliptic curves over finite fields. *Mathematics of Computation*, 76(260):2161–2179, 2007.
- Cer04. Juan Marcos Cervino. On the correspondence between supersingular elliptic curves and maximal quaternionic orders. *arXiv preprint math/0404538*, 2004.
- CG14. Ilya Chevyrev and Steven D Galbraith. Constructing supersingular elliptic curves with a given endomorphism ring. *LMS Journal of Computation and Mathematics*, 17(A):71–91, 2014.
- CH02. Jean-Marc Couveignes and Thierry Henocq. Action of modular correspondences around cm points. In *International Algorithmic Number Theory Symposium*, pages 234–243. Springer, 2002.
- CK19. Leonardo Colò and David Kohel. Orienting supersingular isogeny graphs. *Number-Theoretic Methods in Cryptology 2019*, 2019.
- CL05. Denis Charles and Kristin Lauter. Computing modular polynomials. *LMS Journal of Computation and Mathematics*, 8:195–204, 2005.
- Cou06. Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptology ePrint Archive*, 2006:291, 2006.
- Deu41. Max Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 14(1):197–272, Dec 1941.
- DFFdSG⁺21. Luca De Feo, Tako Boris Fouotsa, Cyprien Delpech de Saint Guilhem, Péter Kutas, Antonin Leroux, Christophe Petit, Javier Silva, and Benjamin Wesolowski. Séta: Supersingular encryption from torsion attacks. In *ASIACRYPT*, 2021.

- DFKL⁺20. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. Sqisign: compact post-quantum signatures from quaternions and isogenies. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 64–93. Springer, 2020.
- E⁺98. Noam D Elkies et al. Elliptic and modular curves over finite fields and related computational issues. *AMS IP STUDIES IN ADVANCED MATHEMATICS*, 7:21–76, 1998.
- EHL⁺18. Kirsten Eisenträger, Sean Hallgren, Kristin Lauter, Travis Morrison, and Christophe Petit. Supersingular isogeny graphs and endomorphism rings: Reductions and solutions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 329–368. Springer International Publishing, 2018.
- Eng09. Andreas Enge. The complexity of class polynomial computation via floating point approximations. *Mathematics of Computation*, 78(266):1089–1107, 2009.
- EPSV23. Jonathan Komada Eriksen, Lorenz Panny, Jana Sotáková, and Mattia Veroni. Deuring for the people: Supersingular elliptic curves with prescribed endomorphism ring in general characteristic. *Cryptology ePrint Archive*, 2023.
- ER01. Friedrich Eisenbrand and Günter Rote. Fast reduction of ternary quadratic forms. In *International Cryptography and Lattices Conference*, pages 32–44. Springer, 2001.
- ES10. Andreas Enge and Andrew V Sutherland. Class invariants by the crt method. In *International Algorithmic Number Theory Symposium*, pages 142–156. Springer, 2010.
- GL23. Eyal Z Goren and Jonathan R Love. On elements of prescribed norm in maximal orders of a quaternion algebra. *arXiv preprint arXiv:2307.16828*, 2023.
- GPS17. Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In *ASIACRYPT*, 2017.
- HS09. Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106. 01 2009.
- JDF11. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 19–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- KLPT14. David Kohel, Kristin E. Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ -isogeny path problem. *IACR Cryptology ePrint Archive*, 2014:505, 2014.
- Koh96. D. Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California at Berkeley, 1996.
- Ler22. Antonin Leroux. *Quaternion Algebra and isogeny-based cryptography*. PhD thesis, Ecole doctorale de l’Institut Polytechnique de Paris, 2022.
- NSV11. Andrew Novocin, Damien Stehlé, and Gilles Villard. An ℓ -reduction algorithm with quasi-linear time complexity. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 403–412, 2011.
- Onu21. Hiroshi Onuki. On oriented supersingular elliptic curves. *Finite Fields and Their Applications*, 69:101777, 2021.

- QKL⁺21. Victoria de Quehen, Péter Kutas, Chris Leonardi, Chloe Martindale, Lorenz Panny, Christophe Petit, and Katherine E Stange. Improved torsion-point attacks on sidh variants. In *CRYPTO*, pages 432–470. Springer, 2021.
- Rob22. Damien Robert. Some applications of higher dimensional isogenies to elliptic curves (overview of results). *Cryptology ePrint Archive*, 2022.
- RS62. J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6:64–94, 1962.
- RS06. Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *Cryptology ePrint Archive*, Report 2006/145, 2006.
- Sch95. René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.
- Sho94. P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Nov 1994.
- Sim05. Denis Simon. Quadratic equations in dimensions 4, 5 and more. *preprint*, 2005.
- Sut11. Andrew Sutherland. Computing hilbert class polynomials with the chinese remainder theorem. *Mathematics of Computation*, 80(273):501–538, 2011.
- Sut12. Andrew V Sutherland. Accelerating the cm method. *LMS Journal of Computation and Mathematics*, 15:172–204, 2012.
- Sut13. Andrew Sutherland. On the evaluation of modular polynomials. *The Open Book Series*, 1(1):531–555, 2013.
- Vél71. J. Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l’Académie des Sciences, Série I*, 273:238–241, juillet 1971.
- Voi18. John Voight. *Quaternion Algebras*. Springer Graduate Texts in Mathematics series, 2018.
- Wat69. William C. Waterhouse. Abelian varieties over finite fields. *Annales Scientifiques de l’E.N.S*, 1969.