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. For that, we revisit 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.

From there, we obtain two main results. First, we show that we can associate these two algorithms with some operations over the quaternion algebra ramified at P and infinity to compute directly Hilbert and modular polynomials $\bmod P$. In that manner, we obtain the first algorithms to compute Hilbert (resp. modular) polynomials modulo P for a good portion of all (resp. all) primes P with a complexity in $\tilde{O}(\sqrt{|D|})$ for the discriminant D (resp. $\tilde{O}(\ell^2)$ for the level ℓ). Due to the (hidden) complexity dependency on P, these algorithms do not outperform the best known algorithms for all prime P but they still provide an asymptotic improvement for a range of prime going up to a bound that is sub-exponential in $|D|$ (resp. ℓ).

Second, we revisit the CRT method for both class and modular polynomials. We show that applying our second heuristic algorithm over supersingular curves to the CRT approach yields the same asymptotic complexity as the best known algorithms based on ordinary curves and we argue that our new approach might be more efficient in practice. The situation appears especially promising for modular polynomials, as our approach reduces the asymptotic cost of elliptic curve operations by a linear factor in the level ℓ. We obtain an algorithm whose asymptotic complexity is now fully dominated by linear algebra and standard polynomial arithmetic over finite fields.

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 cryptograhic 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 results in terms of scaling.

For both class and modular polynomials, the main tools used in the computations are ordinary elliptic curves. The ordinary curves are preferred to supersingular curves because the former have proven over time to lead to more efficient algorithms than the latter. 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).

Contributions. Our main contribution is to reintroduce the use of supersingular elliptic curves in the computation of Hilbert class polynomials and modular polynomials by using the recent progress on the algorithmic Deuring correspondence.

The main sub-routine of our method is aimed 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 OrdersTojInvariantSmall() and it consists merely in a sequential execution of the sub-algorithm (that we call SingleOrderTojInvariant()) from [EHL+18]. SingleOrderTojInvariant() perform the desired translation for one type of maximal order. When everything is done carefully, it can be executed in $O(\log(p)^{4+\varepsilon})$ under experimentally verified heuristics detailed in [KLPT14] and related to the probability for numbers represented by some quadratic form to be prime. Thus, since OrdersTojInvariantSmall() consists in $\#\mathfrak{S}$ executions of Single-OrderTojInvariant(), the total heuristic complexity of OrdersTojInvariantSmall() is $O(\#\mathfrak{S}\log(p)^{4+\varepsilon})$. For a generic p and set of maximal order type $\mathfrak{S}$, we do not know how to do better than that. However, when $\mathfrak{S}$ is close to be maximal (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 OrdersTojInvariant() 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.

A direct application. Our heuristic algorithms OrdersTojInvariantSmall() and OrdersTojInvariant() 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 OrdersTo-jInvariantSmall() and OrdersTojInvariant(). 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 OrdersTojInvariantSmall():

$$O(S\log(P)^{4+\varepsilon} + S\log(S)^{2+\varepsilon}\log(P)).$$

With OrdersTojInvariant(), the complexity becomes

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

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 optimal in both cases: so $O(S\log(P))$.

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 $\mathsf{OrdersTojInvariantSmall}()$.

In comparison, the best previously known generic methods based on the CRT have complexity $O(S^{1+\alpha}\log(S)^{3+\varepsilon})$ where $\alpha = 1$ for Hilbert polynomial and $1/2$ for modular polynomial.

Thus, our algorithm based on $\mathsf{OrdersTojInvariantSmall}()$ will have a better asymptotic complexity when $P = o(2^{S^{\alpha/4}})$. While this is not enough to give an improvement in all cases, this is still an improvement for a significant range of primes P.

When P becomes too big with respect to ℓ, it becomes better to use the CRT method, and we will see that our second algorithm $\mathsf{OrdersTojInvariant}()$ can be applied to this approach as well. The context of CRT typically makes use of several primes that are in $O(S)$ and this is why it will be better to use $\mathsf{OrdersTojInvariant}()$ than $\mathsf{OrdersTojInvariantSmall}()$.

Below, we briefly explain the principle of the CRT method to compute class and modular polynomials, then we detail how to use our new algorithms in that context and outline the differences between our proposed method and the one from Sutherland and BLS.

The CRT for class polynomials. Let us take a prime P and a discriminant $D < 0$. We want an efficient algorithm to compute $H_D(X) \mod P$. Our main algorithm is essentially the same as the one introduced by Sutherland [Sut11].

Let us write $\mathfrak{O}$ for the quadratic imaginary order of discriminant $D < 0$. We give a brief outline of Sutherland's algorithm. We may assume that the factorization of D is known as computing it is negligible compared to the rest of the computation. We define $\mathcal{P}_D$ 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 $\mathcal{P}_D$ with $\prod_{i=1}^n p_i > 2^{B_D}$.
2. Compute a suitable representation of $\mathrm{Cl}(D)$.
3. For each $p_i \in \mathcal{P}_D$:
   (a) Compute the coefficients of $H_D \mod p_i$.
   (b) Update CRT sums for each coefficient of H_D.
4. Recover the coefficients of $H_D \mod P$.

The only difference with the concrete method proposed by Sutherland and ours is in the choice of the set $\mathcal{P}_D$. In [Sut11], the set $\mathcal{P}_D$ is made of primes of the form $(t^2 - Dv^2)/4$, whereas in our case $\mathcal{P}_D$ is made of non-split primes that are coprime with the square part of D (note that those two conditions are mutually exclusive).

In both cases, the $H_D \mod p_i$ 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.

For the supersingular case, we recover the roots using our second algorithm OrdersTojInvariant(). We will see that, for the CRT, the size of the primes p_i is such that we are in the regime that favours OrdersTojInvariant() over OrdersTo-jInvariantSmall().

Note that the case of non-split prime p_i have already been considered in the context of the CRT method by [BBEL08], but only for very small primes because of the inefficiency of Cervino's algorithm.

For modular polynomials. Let ℓ be a prime distinct from P. We want an efficient algorithm to compute $\Phi_\ell(X, Y) \mod P$. It can be done in a very similar fashion to class polynomials: compute $\Phi_\ell \mod p_i$ for some p_i in a set $\mathcal{P}_\ell$ and reconstruct the final polynomial with CRT.

The idea introduced by Bröker, Lauter and Sutherland (BLS in the rest of this article) is to use primes of the form $(t^2 - 4v^2 D\ell^2)$ with $t, v, D \in \mathbb{N}$ for which there is a very specific volcano structure. 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 $\mathfrak{O}$ for some quadratic imaginary order $\mathfrak{O}$ of discriminant D and class number bigger than $\ell + 2$ and the curves with endomorphism ring isomorphic to $\mathbb{Z} + \ell\mathfrak{O}$. Since the latter are ℓ-isogenous to the former, it is possible to recover the full $\Phi_\ell \mod p_i$ 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 Vélu formulas).

We prove that using the Deuring correspondence and OrdersTojInvariant(), 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.

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 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* $\varphi : E_1 \to E_2$ 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 Vélu formulas [Vél71] can be used to compute any cyclic isogeny from its kernel. For any $\varphi : E_1 \to E_2$, there exists a unique dual isogeny $\hat{\varphi} : E_2 \to E_1$, satisfying $\varphi \circ \hat{\varphi} = [\deg(\varphi)]$.

Endomorphism ring. An isogeny from a curve E to itself is an *endomorphism*. The set $\mathrm{End}(E)$ of all endomorphisms of E forms a ring under addition and composition. For elliptic curves defined over a finite field $\mathbb{F}_q$, $\mathrm{End}(E)$ 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 $\pi : E \to E^p$. For any supersingular curve E, the property $\mathrm{End}(E) \cong \mathrm{End}(E^p)$ 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 $a, b \in \mathbb{Q}^\star$ 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 $\alpha = a_1 + a_2 i + a_3 j + a_4 k$ to its conjugate $\overline{\alpha} = a_1 - a_2 i - a_3 j - a_4 k$. We define the *reduced trace* and the *reduced norm* by $tr(\alpha) = \alpha + \overline{\alpha}$ and $n(\alpha) = \alpha\overline{\alpha}$.

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* $\mathrm{Typ}\, \mathcal{O}$ *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 $\overline{I}$ is the set of conjugates of elements of I, which is an ideal satisfying $I\overline{I} = n(I)\mathcal{O}_L(I)$ and $\overline{I}I = n(I)\mathcal{O}_R(I)$.

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 $\beta \in \mathcal{B}^\star$ such that $\beta\mathcal{O}_1 = \mathcal{O}_2\beta$. Two left $\mathcal{O}$-ideals I and J are equivalent if there exists $\beta \in \mathcal{B}^\star$, 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 equivalences classes of left $\mathcal{O}$-ideals, and we denote the set of such classes by $\mathrm{Cl}(\mathcal{O})$.

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 $E \to \mathrm{End}(E)$. 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 $\varphi : E_0 \to E_1$ is an isogeny, is sent to a left integral $\mathcal{O}_0$-ideal. The right order of this ideal is isomorphic to $\mathrm{End}(E_1)$. 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

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

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.

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).

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

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

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 algorithm.

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 Section 3.2 target two opposite situations with respect to the relative size of p and $\mathfrak{S}$. The first algorithm is called Orders-TojInvariantSmall(). As the name suggests, it targets the case where $\#\mathfrak{S}/p$ is "small", and it is a quite direct application of standard results on the effective Deuring correspondence. Indeed, it consists in the sequential execution for each element in $\mathfrak{S}$ of the algorithm SingleOrderTojInvariant() introduced in [EHL+18] and that can compute the j-invariants associated to one maximal order type. OrdersTojInvariantSmall() will work the best when $\mathfrak{S}$ is made of a "small" portion of all possible types. Its asymptotic complexity is $O(\#\mathfrak{S} \log(p)^{4+\varepsilon} + \log(p)^{6+\varepsilon})$ and works for any prime p and set $\mathfrak{S}$.

The second algorithm is called OrdersTojInvariant() 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 $p/\#\mathfrak{S} = \Theta(\log(p)^{1+\varepsilon})$.

In the rest of this work, we will sometimes refer to these two algorithms by calling them the "small" and "big" case respectively.

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 $\operatorname{Typ}\mathcal{O}$ 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. Let us take a maximal order $\mathcal{O}$. It can be shown (see [CG14]) that $1, x_1, x_2, x_3$ is a basis of $\mathcal{O}$ where $2x_1 - tr(x_1), 2x_2 - \operatorname{tr}(x_2), 2x_3 - \operatorname{tr}(x_3)$ realize the successive minima of the lattice $\mathcal{O}^T = \{2x - \operatorname{tr}(x) | x \in \mathcal{O}\}$. Thus, if we take the Gram matrix of this basis (possibly reordering x_1, x_2, x_3 so that $\operatorname{tr}(x1x_2) \leq \operatorname{tr}(x1x_3) \leq \operatorname{tr}(x_2x_3)$ if there are some equalities of norm between x_1, x_2 and x_3) we obtain 10 values (the Gram matrix is symmetric) that uniquely represent the lattice $\mathcal{O}$.

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

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 of $\mathcal{O}^T$.
3. Derive the basis of $\mathcal{O}$.
4. Compute the Gram Matrix M associated to this basis.
5. Output $H(\mathcal{O}) = h((M_{i,j})_{1 \leq i \leq j \leq n})$.

Proposition 1. *The hash function H presented above can be computed in $O(\log(X)^{1+\varepsilon})$ 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 using the algorithm to reduce ternary quadratic form of [ER01], or the $\tilde{L}^1$ algorithm from [NSV11] to perform lattice reduction in small dimension, to compute the successive minimas of $\mathcal{O}^T$.

3.2 Matching maximal orders and j-invariant

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 case of small $\mathfrak{S}$. The algorithm in the small case is quite simple to describe, it is made of consecutive executions of a sub-algorithm $\mathsf{SingleOrderTojInvariant}()$ described as [EHL+18, Algorithm 12]. The complexity of this algorithm was analyzed by Galbraith, Petit and Silva in [GPS17] and it is $O(\log(p)^{6+\varepsilon})$ for the first execution and $O(\log(p)^{4+\varepsilon})$ after that.

The algorithm we obtain by applying $\mathsf{SingleOrderTojInvariant}()$ on each element of the input set $\mathfrak{S}$ is called $\mathsf{OrdersTojInvariantSmall}()$ and its complexity is $O(\#\mathfrak{S}\log(p)^{4+\varepsilon} + \log(p)^{6+\varepsilon})$. Note that this result holds under various heuristic that are detailed in [KLPT14,GPS17]. In terms of space, the complexity is optimal: $O(\#\mathfrak{S}\log(p))$.

The case of big $\#\mathfrak{S}$. The nice thing with $\mathsf{SingleOrderTojInvariant}()$ is that it allows us to target specifically any maximal order type. This flexibility makes $\mathsf{OrdersTojInvariantSmall}()$ a good candidate when $\mathfrak{S}$ is not too big, but it becomes more and more costly as $\#\mathfrak{S}/p$ increases.

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 will be able to do this more efficiently than applying $\mathsf{SingleOrderTojInvariant}()$ for each 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 $\mathrm{Typ}\,\mathcal{O}_0$ 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 $\mathsf{OrdersTojInvariant}()$. 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. Note that our heuristic is different from the ones used in the analysis of $\mathsf{SingleOrderTojInvariant}()$.

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 $C_0, 1 < C_1 < C_2$ such that for every number $N > B$, there exists a set of n coprime factors $L_1, \cdots, L_n$ with $\sqrt{C_0 \log(N)} \leq L_i < C_0 \log(N)$ for all $1 \leq i \leq n-1$, $C_0/2 \log(N) \leq L_n < C_0 L_n$ and $C_1 N \leq \prod_{i=1}^{n} \Phi(L_i) < C_2 N$.*

The proof of Lemma 1 is not hard but it is quite tedious so it is given in Appendix A.

Remark 1. In practice, for a given N we will call such a set of factors $L_1, \cdots, L_n$, a *degree-basis* for N. When each L_i is a prime power of the $(n-i+1)-th$ smallest prime number ℓ_{n-i+1}, we say that $L_1, \ldots, L_n$ is a **good** *degree-basis* of N. The choice of sorting the prime factors of the L_i in descending order will be important for our algorithm.

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

Proposition 2. *Take an integer $N > B$ with good degree-basis $L_1, L_2, \cdots, L_n$ with $n > 6$. Then, there exists a constant $k \leq 6$ and a constant C_3 such that*

$$\sum_{i=1}^{n-k} (n-i) \prod_{j=1}^{i} \Phi(L_j) < C_3 \frac{N}{\log(N)^2} \tag{3.1}$$

Proof. We are going to show the bound for $k = 6$.

For that we are going to use the equalities $\sum_{i=0}^{m} X^i = \frac{X^{m+1}-1}{X-1}$ and $\sum_{i=1}^{m} iX^i = X(\frac{mX^{m+1}-(m+1)X^m+1}{(X-1)^2})$ for any m and X. By our definition of a degree-basis in Lemma 1, we have that $\Phi(L_j) \geq L_i \geq \sqrt{C_0 \log(N)}$ 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 $X = \sqrt{C_0 \log(N)}$. We have $\sum_{i=1}^{n-6} (n-i) \prod_{j=1}^{i} \Phi(L_i) < C_2 N \sum_{i=1}^{n-6} (n-i) X^{n-i}$. Then, we have $\sum_{i=1}^{n-6} (n-i) X^{n-i} = \sum_{i=6}^{n-1} iX^i = X^6 \sum_{i=0}^{n-7} (i+6) X^i$. Then, we apply our two equalities to get $\sum_{i=1}^{n-6} (n-i) X^{n-i} = X^6 (6 \frac{X^{n-6}-1}{X-1}) + \frac{(n-7)X^{n-5}-(n-6)X^{n-6}+X}{(X-1)^2}) = X^6 \frac{(n-1)X^{n-5}-(n-12)X^{n-6}-5X+1}{(X-1)^2}$. 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, $\sum_{i=1}^{n-6} (n-i) \prod_{j=1}^{i} \Phi(L_i) < \frac{C_2 C_3'}{C_0^2} \frac{N}{\log_2(N)}$.

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 2. 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 OrdersTojInvariant(), we assume the knowledge of a good degree-basis $L_1, \cdots, L_n$ (see Remark 1) of $C_4 p$ (where C_4 is the constant in Claim 1). In OrdersTojInvariant(), 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, the torsion subgroup $E_0[N]$ is defined over an extension $\mathbb{F}_{p^{2m}}$ of degree $m \leq N$ over $\mathbb{F}_{p^2}$.*

Algorithm 1 OrdersTojInvariant()

```
Input: A prime $p$. A good degree-basis $L_1, \cdots, L_n$, 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: Compute a supersingular curve $E_0$ over $\mathbb{F}_{p^2}$ with known endomorphism ring $\mathcal{O}_0$.
 2: Compute $I_0 = \mathcal{O}_0\langle\alpha_0, L\rangle$ one left integral $\mathcal{O}_0$-ideal of norm $L$.
 3: Find $\alpha \in \mathrm{End}(E_0)$ such that $\gcd(n(x+y\alpha), L) = 1$ for all $x, y$ with $\gcd(x, y, L) = 1$.

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

Proposition 3. *Assuming Claim 1,* $\mathsf{OrdersTojInvariant}(p)$ *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 $\varphi_n \circ \ldots \circ \varphi_1$ 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 $\varphi_{i-1} \circ \cdots \circ \varphi_1$ 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 18.

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

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 $\ker \alpha_0(C_j + D_j\overline{\alpha}) \cap E_0[L_j] = \langle [C_j]R_{0,j} + [D_j]S_{0,j}\rangle$.

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 $([C_j] + [D_j]\overline{\alpha})([C_j]R_{0,j} + [D_j]S_{0,j}) = ([C_j] + [D_j](\overline{\alpha})([C_j] + [D_j]\alpha)(R_{0,j}) = [n(C_j + D_j\alpha)]R_{0,j}$. This proves that we have $\langle [C_j]R_{0,j} + [D_j]S_{0,j}\rangle \subset \ker \alpha_0(C_j + D_j\overline{\alpha}) \cap E_0[L_j]$.

By definition of α the scalar $n(C_j + D_j\alpha)$ 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 $\ker \alpha_0(C_j + D_j\overline{\alpha}) \cap E_0[L_j] = \langle [C_j]R_{0,j} + [D_j]S_{0,j}\rangle$.

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 log factor basis used is bigger than 6 so we can apply Proposition 2.

Proposition 4. *Under GRH, Step 1 can be executed $O(\mathsf{poly}(\log(p)))$.*

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 supersigular 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 Vélu formulae. Since the suborder $\langle 1, \iota, \pi, \iota \circ \pi \rangle$ has an index in $O(d) = O(\mathsf{poly}(\log(p)))$ inside $\mathrm{End}(E_0)$, and recovering the full endomorphism ring can be done in $O(\mathsf{poly}(\log(p)))$.

This proves the result for Step 1.

Proposition 5. *Step 2 and Step 3 can be executed in $O(\mathsf{poly}(\log(p)))$ and the output α_0 and α have coefficients in $O(\mathsf{poly}(p))$ 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 $O(p)$.

The Steps 2 and 3 can be solved in a similar manner despite a final goal that is quite different. For Step 2, it is sufficient to find an β_0 such that 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 3, a sufficient condition to obtain α such that $\gcd(n(x+y\alpha), L) = 1$ 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 $X^2 + \mathrm{tr}(\beta)X + n(\beta) \mod L_i$.

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 4 can be executed in* $O(\mathsf{poly}(\log(p)))$.

Proof. The loop is repeated $O(\log(p))$ times. Computing a basis is a very standard task and it can be done in $O(\mathsf{poly}(\log(p)))$ since $m_i = O(\log(p))$. The generator $R_{0,i}$ can be computed in $O(\mathsf{poly}(\log(p)))$ 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 $\mathrm{End}(E_0)$ and then performing the scalar multiplications corresponding to the coefficients of α in this basis. The first part can be done in $O(\mathsf{poly}(\log(p)))$ by choice of E_0 and the second can also be done in $O(\mathsf{poly}(\log(p)))$ by the size bound given on the coefficients of α in Proposition 5.

The main computational task of $\mathsf{OrdersTojInvariant}()$ is quite clearly performed during the loop in line 14. 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 24 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 24 produces e_i polynomials of degree ℓ_i over $\mathbb{F}_{p^2}$ in $O(e_i^2)$ operations over $\mathbb{F}_{p^{m_n}}$. These polynomials uniquely defines the isogeny φ.*

Proof. It can be shown with the Vélu 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 25,26 of the FOR loop in line 18 is upper bounded by $C(n - i)\log(p)M_{\mathbb{P}}(\log(p))$. When $i > n - 6$, we have the upper-bound $C(n - i)\mathrm{llog}(p)^2 M_{\mathbb{P}}(\log(p))$*

Proof. For each $i < j \leq n$, we need to evaluate the polynomial produced by Step 24 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 $e_i = O(\operatorname{llog}(p))$ polynomials of degree $\ell_i = O(1)$ and we get the desired result.

Remark 3. The Vélusqrt 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 18 is upper bounded by $C\Phi(L_i)(n-i)\log(p)M_{\mathbb{P}}(\log(p))M_{\mathbb{Z}}(\log(p))$. When $n-6 < i < n$, the number of binary operations is smaller than $C\Phi(L_i)\operatorname{llog}(p)^2 M_{\mathbb{P}}(\log(p))M_{\mathbb{Z}}(\log(p))$.*

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

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

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

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

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 16 is upper bounded by $C\prod_{j=1}^{i}\Phi(L_i)(n-i)\log(p)^{3+\varepsilon}$. When $n-6 < i < n$, it is upper bounded by $C\prod_{j=1}^{i}\Phi(L_i)(n-i)\log(p)^{2+\varepsilon})$. 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 $\prod_{j=1}^{i-1}\Phi(L_i)$ 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 $\prod_{j=1}^{n}\Phi(L_i) = O(p)$ 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 $O(\text{llog}(p)^2 M_{\mathbb{P}}(\log(p)) M_{\mathbb{Z}}(\log(p)))$ and this proves the result.

Proposition 11. *The loop in line 14 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 $C\sum_{i=1}^{n-6}(n-i)\prod_{j=1}^{i}\Phi(L_i)\log(p)^{3+\varepsilon}$. So we get $O(p\log(p)^{1+\varepsilon})$ after applying Proposition 2.

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

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

Since by Lemma 1, $L_n > C_0/2\log(p)$, we have $\prod_{j=1}^{i}\Phi(L_i) \leq C'p/\log(p)$ 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 $\mathfrak{S}$,* OrdersTojInvariant() *can be executed in*

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

binary operations.

Remark 4. Similarly, we can show that the space requirement of OrdersTojInvariant() is in $O(\#\mathfrak{S}\log(p) + p\log(p))$.

Good choice of primes. The analysis we provided above for both OrdersTojInvariantSmall() and OrdersTojInvariant() 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 OrdersTojInvariant() 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 OrdersTojInvariant().

4 Computation of the Hilbert class polynomial

Let D be a negative discriminant. As explained in the introduction, the CRT method (whether it is to compute H_D in $\mathbb{Z}$ or modulo a big prime P) consists mainly in computing $H_D \mod p_i$ for a bunch of small primes p_i.

In Section 4.1, we introduce an algorithm to perform that computation when the prime p_i is such that the roots of H_D over $\overline{\mathbb{F}_p}$ are j-invariants of supersingular curves. This algorithm uses the algorithm $\mathsf{OrdersTojInvariant}()$ from Section 3.2 as its main sub-routine. This algorithm has the best known complexity for a generic prime when P is "small" (relatively to D). However, if the prime is big, it will be more efficient to use $\mathsf{OrdersTojInvariantSmall}()$ instead as explained in Section 3. When modified to use $\mathsf{OrdersTojInvariantSmall}()$ instead of $\mathsf{OrdersTojInvariant}()$, the algorithm we present in Section 4.1 can also compute directly $H_D \mod P$ for any primes P that is not split and coprime with the square factor of D, and its complexity is in $O(h(D)\log(P)^{4+\varepsilon})$.

Depending on the respective size of P and D (in particular when $\log(P) = o(|D|^{1/8})$), this algorithm has a better asymptotic complexity than all known algorithm.

In Section 4.2, we compare the gain of using our algorithm with supersingular curves in the CRT method over the best known algorithm from Sutherland [Sut11] based on ordinary curve.

4.1 Computing the class polynomial modulo a prime.

Let us fix some prime p and some negative discriminant D. We write h for the class number of D.

Our goal in this section, is to explain how to compute the Hilbert class polynomial $H_D(X) \mod p$. In all cases, this polynomial is reconstructed from its roots. When p can be written as $(t^2 - Dv^2)/4$ for two integers t, v, the prime p is split in the quadratic order $\mathfrak{O}$ of discriminant D, and the roots of H_D are j-invariants of ordinary curves in $\mathbb{F}_p$. This case was treated by Sutherland in [Sut11]. In the opposite case where p is non-split and coprime with the square part of D, the roots are j-invariants of supersingular curves over $\mathbb{F}_{p^2}$ and we explain below how to compute them.

In the ordinary case, the interesting curves are obtained in two main steps: start by identifying one interesting curve, and then enumerate through all the interesting curves using the action of the class group $\mathrm{Cl}(\mathfrak{O})$.

For supersingular curves, we have three main steps. We start to do something very similar to what is done for ordinary curves, but working over the maximal orders of the quaternion algebra $\mathcal{B}_{p,\infty}$. The idea is that this step is much more efficient when working directly over $\mathcal{B}_{p,\infty}$ because the operations are much simpler. 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 constitue a set $\mathfrak{S}_D(p)$ of maximal order type and we can apply $\mathsf{OrdersTojInvariant}()$ 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.

At a high level, our algorithm works in the following way :

1. Find a maximal order $\mathcal{O}$ in $\mathcal{B}_{p,\infty}$ with $\mathfrak{O} \hookrightarrow \mathcal{O}$.
2. Use the action of $\mathrm{Cl}(D)$ to find $\mathfrak{S}_D(p)$, the set of isomorphism classes of maximal orders in $\mathcal{B}_{p,\infty}$ with an optimal embedding of $\mathfrak{O}$ and record their multiplicities.
3. Compute the roots of $H_D \mod p$ with $\mathsf{OrdersTojInvariant}(p, \mathfrak{S}_D(p))$.
4. Recover $H_D \mod p$.

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 Séta 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 once the factorization of the determinant is known. In our case, the quadratic form we consider is basically $b, c, d, e \mapsto (qb^2 + p(c^2 + qd^2) - e^2 D$ and its determinant is equal to $p^2 q^2 D 2^f$ for some small integer f. Hence, the full factorization is easy to compute because we know the factorization of D.

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 $A = O(\mathsf{poly}(\log(pD)))$. 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 $O(\mathsf{poly}(\log(p|D|)))$.

Hence, this step can be performed in $O(\mathsf{poly}(\log(p|D|)))$ 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]. 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], 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 $\mathsf{OrdersTojInvariant}()$ 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 $\mathsf{OrdersTojInvariantSmall}()$ algorithm and obtain a complexity of $O(\sqrt{|D|} \log(|D|)^{\varepsilon} \log(p)^{4+\varepsilon})$.

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 that, when using the OrdersTojInvariant() algorithm, the complexity is

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

With OrdersTojInvariantSmall(), the complexity becomes

$$O(\sqrt{|D|}(\log(|D|)^{\varepsilon}\log(p)^{4+\varepsilon}+\log(|D|)^{2+\varepsilon}\log(p)^{1+\varepsilon})).$$

Thus, we see that the first algorithm will be better for small values of p. There will be a cut-off around a value of p in $O(\sqrt{|D|}\log(|D|)^{3+\varepsilon})$. In that range of prime, our algorithm with OrdersTojInvariant() has the best known generic complexity. For primes, bigger than that, it is better to use the variant with OrdersTojInvariantSmall() due to the quasi-linear dependency in p. 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 $O(|D|^{1+\varepsilon})$) will happen for $p=O(2^{|D|^{1/8}})$.

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

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

In this section, we analyze the benefit of using supersingular curves in the CRT method compared to ordinary curves as done by Sutherland in [Sut11]. We refer the reader to the algorithm outlined in Section 1.1.

The choice of primes. The final crucial parameter for stating a complexity estimate for the CRT method with the algorithm of Section 4.1 is the choice of primes $p_1,\dots,p_n\in\mathcal{P}_D$. In fact, for supersingular curves, this part is quite easy. It suffices to take all non-split primes coprime with the square part of D. For practical efficiency, some primes might be better than others so it might be worth considering a finer metric than size, but for a first and simple estimate, it is easier to consider that we take the n smallest primes satisfying the reduosity constraint. Under GRH, it can be shown that we have $n=O(B_D/\log(B_D))$ and $\max_{1\leq i\leq n}p_i=O(\log(B_D))$. With the usual $B_D=O(\sqrt{(|D|)}\log(|D|))$ that holds under GRH, we get that we can take $O(\sqrt{|D|})$ primes with $\max_{1\leq i\leq n}p_i=O(\sqrt{|D|}\log(|D|))$.

The final heuristic complexity estimate. Thus, under GRH and the heuristic of Section 3, if we sum the complexity estimate given in Section 4.1 over all p_i; the final complexity estimate of the CRT method with supersingular curves is $O(|D|\log(|D|)^{3+\varepsilon})$. This is the same asymptotic complexity as the CRT method for ordinary curves introduced by Sutherland [Sut11]. The dominant step is also the same: the polynomial reconstruction (Step 4 in our algorithm). However, note that in practice this part might not be the bottleneck due to better hidden constants.

Comparison of supersingular and ordinary cases. Let us start by the reconstruction step. We remind that the concrete complexity of this step is $O(\sqrt{|D|}\text{-}\log(|D|)^{2+\varepsilon}\log(p)$. It is pretty similar in both cases 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 $H_D \mod p_i$ 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 $\mathbb{F}_{p_i^2}$ as in the ordinary case (and thus avoid the constant overhead brought by multiplications over $\mathbb{F}_{p_i}$). 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 $\mathsf{OrdersTojInvariant}()$ in Step 3, which has a $O(|D|\log(|D|)^{2+\varepsilon})$ 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 $O(|D|\log(|D|)^{1+\varepsilon}$ for ordinary curves). Moreover, we can take all the small primes that satisfy the reduosity 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 log-factor-basis used for each of those prime in order to minimize the degree of the extension required to compute the isogenies in OrdersTojInvariant(). 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 OrdersTojInvariant() 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. OrdersTojInvariant() can be easily modified to handle several sets $\mathfrak{S}_1, \dots, \mathfrak{S}_k$ more efficiently than k executions of OrdersTojInvariant() for each $\mathfrak{S}_i$.

Thus, if we have several discriminants $D_1, \dots, D_k$, and a prime p in $\bigcap_{1 \leq i \leq k} \mathcal{P}_{D_k}$. A good part of the computations performed to compute $H_{D_1}, \dots, H_{D_k} \mod p$ 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

Let ℓ be a prime number. As explained in the introduction, the CRT method (whether it is to compute Φ_ℓ in $\mathbb{Z}$ or modulo a big prime P) consists mainly in computing $\Phi_\ell \mod p_i$ for a bunch of small primes p_i.

In Section 5.1, we introduce an algorithm to do so for any big enough prime p_i. This algorithm uses the algorithm OrdersTojInvariant() from Section 3.2 as its main sub-routine. And it has the best known complexity when P is "small" (relatively to ℓ) However, if the prime is big, it will be more efficient to use OrdersTojInvariantSmall() as explained in Section 3. When modified to use Orders-TojInvariantSmall() instead of OrdersTojInvariant()), the algorithm we present in Section 5.1 can also compute directly $\Phi_\ell \mod P$ for any prime P and its complexity is in $O(\ell^2 \log(P)^4)$. Depending on the respective size of P and ℓ (in particular when $\log(p) = o(\ell^{1/4})$), this algorithm has a better asymptotic complexity than all known algorithm. Note that this method works for every p and ℓ (as soon as p is big enough). In particular, it can be used to compute $\Phi_\ell \mod p$ in applications where we need to find ordinary curves that are ℓ-isogenous. This was not the case for Hilbert polynomial where the roots are either all ordinary or all supersingular.

In Section 5.2, we compare the gain of using our algorithm with supersingular curves in the CRT method over the best known algorithm from Broker, Lauter and Sutherland [BLS12] based on ordinary curve.

5.1 Computing modular polynomial modulo a prime

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 Φ_ℓ is interpolated from $\Phi_\ell(j, Y)$ for enough j-invariants j. The univariate polynomial $\Phi_\ell(j, Y)$ is reconstructed from its roots, that are the j-invariants of curves ℓ-isogenous to j.

Once again, we start by identifying the interesting curves through the Deuring correspondence. More concretely, for a set of prescribed maximal orders, we will compute all the ℓ-ideals associated to these maximal orders and compute their right orders. Then, we apply $\mathsf{OrdersTojInvariant}()$ to find the j-invariants corresponding to the maximal orders computed during the previous steps, finally we interpolate the modular polynomial $\Phi_\ell \mod p$.

Here is how it can be done concretely:

1. Compute a set of maximal order types $\mathcal{O}_{1,0}, \mathcal{O}_{2,0}, \ldots, \mathcal{O}_{m,0} \subset \mathcal{B}_{p,\infty}$ for $m \geq \ell + 2$.
2. For each $1 \leq i \leq m$, compute the types $\mathcal{O}_{i,1} \ldots \mathcal{O}_{i,\ell+1}$ of maximal orders connected to $\mathcal{O}_{i,0}$ with an ideal of norm ℓ.
3. Create the set $\mathfrak{S}_\ell$ made of the hashed values of all types in $\bigcup_{1\leq i\leq m, 1\leq k\leq \ell+1} \mathcal{O}_{i,k}$.
4. Compute each j-invariant associated to the elements of $\mathfrak{S}$ with $\mathsf{OrdersTojInvariant}(p, \mathfrak{S}_\ell)$.
5. For each $1 \leq i \leq m$ compute $\Phi_\ell(j_{i,0}, X)$ from its roots $j_{i,k}$ for $1 \leq k \leq \ell + 1$.
6. Reconstruct $\Phi_\ell(X, Y) \mod p$ from the $\Phi_\ell(j_{i,0}, X)$.

With what we saw in Section 4 all the steps of the algorithm above are pretty straightforward. Note that we have $m = O(\ell)$ and we assume that p is big enough so that there exists more than m maximal order types over $\mathcal{B}_{p,\infty}$. We briefly recall the complexities of each step.

1. By Claim 1, the complexity is $O(\ell \log(p)^{1+\varepsilon})$ by computing the m distinct types $\mathcal{O}_{i,0}$ as right orders of L-ideals for some $L = O(p)$ (computing right orders can be done in $O(\log(p)^{1+\varepsilon})$ in that case).
2. $O(\ell^2(\log(p) + \log(\ell))$ as there are $(\ell + 1)m$ ideal computation and each one takes $O((\log(p) + \log(\ell))^{1+\varepsilon})$.
3. $O(\min(p, \ell^2)(\log(p)+\log(\ell))^{1+\varepsilon})$ as there are at most $O(\min(p, \ell^2))$ maximal order types in $\mathfrak{S}_\ell$ and it takes $O(\log(p) + \log(\ell))^{1+\varepsilon})$ to compute their hash with the function H.
4. $O(p \log(p)^{1+\varepsilon} + \min(p, \ell^2) \log(p)^{2+\varepsilon})$ as $\#\mathfrak{S}_\ell = O(\min(p, \ell^2))$.
5. $O(\ell^2 \log(\ell)^{2+\varepsilon} \log(p)^{1+\varepsilon})$ as shown in [BLS12].
6. $O(\ell^2 \log(\ell)^{2+\varepsilon} \log(p)^{1+\varepsilon})$ as shown in [BLS12].

Total complexity. In conclusion, the complexity of the algorithm to compute $\Phi_\ell \mod p$ with $\mathsf{OrdersTojInvariant}()$ is

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

When using OrdersTojInvariantSmall() instead, we obtain the following asymptotic complexity:

$$O(\ell^2(\log(p)^{4+\varepsilon} + \log(\ell)^{2+\varepsilon}\log(p)^{1+\varepsilon})),$$

Thus, we see that the first algorithm based on OrdersTojInvariant() 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 OrdersTojInvariant() has the best known generic complexity to compute $\Phi_\ell \mod p$. For primes bigger than that, it is better to use the variant with OrdersTojInvariantSmall() to avoid the quasi-linear dependency in p. 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 $O(\ell^{3+\varepsilon})$) will happen for $p = O(2^{\ell^{1/4}})$.

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

5.2 Applications to the CRT method and comparison with existing method

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 OrdersTojInvariant().

Choice of primes. The only constraint on our CRT primes p_i is the size. We need to have enough maximal order types, which means that $p/12$ must be slightly bigger than ℓ. As for Hilbert polynomials, we analyze the case where we simply select the set of primes $\mathcal{P}_\ell$ as made of the smallest primes satisfying the constraint and such that $\prod_{i=1}^{n} p_i > 2^{B_\ell}$ where B_ℓ is the bound on the bit-size of the coefficients of $\Phi_\ell(X,Y)$. In practice, it might be better to select primes that will enable an efficient execution of OrdersTojInvariant() but this is harder to analyze. Since we have $B_\ell = O(\ell\log(\ell))$, we expect to have $n = O(\ell)$ primes with $\max_{1\leq i\leq n} p_i = O(\ell\log(\ell))$.

The final heuristic complexity estimate. With everything we said above, under GRH and the heuristic of Section 3.2, the asymptotic cost of the CRT method with our algorithm is $O(\ell^3\log(\ell)^{3+\varepsilon})$. This is the same as the method based on ordinary curves. In both cases, the asymptotic bottleneck is the polynomial reconstruction step.

Practical comparison between supersingular and ordinary cases. Let us start by the reconstruction step. We remind the reader that the complexity of this step is $O(\ell^2\log(\ell)^{2+\varepsilon}\log(p))$. As for the class polynomial case, these two steps are pretty similar in the supersingular and ordinary cases. Similarly, we expect the practical cost to be the same. It is less easy to see in the modular polynomial case, but the symmetry of Φ_ℓ with respect to the action of the Galois group of $\mathbb{F}_{p^2}$ allows us to perform almost all computations over $\mathbb{F}_{p_i}$. The linear dependency in $\log(p_i)$ concludes the claim.

For the rest, we expect our algorithm to outperform the BLS method. There are various explanations behind that claim, but all of them are basically implications of the following fact: we can consider primes $p_i \in \mathcal{P}_\ell$ with $p_i = O(\ell \log \ell)$. For each p_i, there are $O(p_i)$ supersingular curves (and so $O(p_i)$ supersingular j-invariants over $\mathbb{F}_{p^2}$), which is enough to reconstruct the modular polynomial Φ_ℓ, even though intuitively, it should require $O(\ell^2)$ distinct points.

The first implication of that fact, is that we drastically reduce the asymptotic cost of the expensive elliptic curve operations in the execution of the CRT method. Indeed, with supersingular curves, this part only requires $O(\ell^{2+\varepsilon})$ operations over various $\mathbb{F}_{p_i}$ (against $O(\ell^{3+\varepsilon})$ in BLS).

The polynomial reconstruction should also be positively impacted by the fact that the $O(\ell)$ univariate polynomials required to interpolate Φ_ℓ have a lot of common roots.

Moreover, if we remove the common polynomial reconstruction part, we see that the asymptotic cost is also in favour of the supersingular case with $O(\ell^3 \log(\ell)^{1+\varepsilon})$ against $O(\ell^3 \log(\ell)^{3+\varepsilon})$ in BLS. And the hidden constants should also be in favour of supersingular case since the dominant step in our algorithm consists in basic operations over lattices in the quaternion algebra $\mathcal{B}_{p,\infty}$ while these are $\mathbb{F}_p$ operations in BLS.

All in all, it is very likely that using our algorithm in the CRT method will bring a practical improvement over the BLS algorithm. Contrary to the class polynomial case, where we need to be more careful, we do not expect that it could be favourable to use a mix of supersingular and ordinary curves instead of supersingular curves only. However, this improvement will be less and less noticeable as the value of ℓ increases, since the asymptotically dominant step is the polynomial reconstruction which is the same in both methods.

Batching the computation. Similarly to the modular polynomial case, the set of small primes can be reused in the computations over various ℓ. Once again, 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.

On the computation of $\Phi_\ell(E, Y)$. In an application like the SEA algorithm, computing the full Φ_ℓ is actually useless. What we need to do is to evaluate $\Phi_\ell(E, Y)$ for some curve E defined over $\mathbb{F}_p$. Sutherland [Sut13] showed how to adapt the CRT method to that purpose. The complexity is essentially the same as the one to compute the full polynomial, but the memory requirement is smaller and the practical complexity is better as well.

Thus, our new algorithm yields an improvement for that task as well (when p is not too big). However, for a generic E (that can be ordinary for instance), it is not clear that we can do better than the complexity to compute the full Φ_ℓ. We have an obvious improvement to $O(\ell \log(p)^4)$, when E is a supersingular curve of known endomorphism ring but this case is not so useful for the usual applications (in that case, we can simply apply the Deuring correspondence to circumvent the need for the modular polynomial).

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. When used directly, we see that we obtain the first algorithms of complexity in ℓ^2 and $\sqrt{|D|}$ for generic primes. Depending on the relative size of ℓ,$|D|$ 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 the version with ordinary curves, 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. *E*xperimental Mathematics, pages 1–28, 2021.

AM93. A Oliver L Atkin and François Morain. Elliptic curves and primality proving. *M*athematics of computation, 61(203):29–68, 1993.

BBEL08. Juliana Belding, Reinier Bröker, Andreas Enge, and Kristin Lauter. Computing hilbert class polynomials. In *I*nternational 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. *A*NTS, 2020.

BLS12. Reinier Bröker, Kristin Lauter, and Andrew Sutherland. Modular polynomials via isogeny volcanoes. *M*athematics of Computation, 81(278):1201–1231, 2012.

BS07. Reinier Bröker and Peter Stevenhagen. Efficient cm-constructions of elliptic curves over finite fields. *M*athematics of Computation, 76(260):2161–2179, 2007.

Cer04. Juan Marcos Cervino. On the correspondence between supersingular elliptic curves and maximal quaternionic orders. *a*rXiv 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.

ER01. Friedrich Eisenbrand and Günter Rote. Fast reduction of ternary quadratic forms. In *International Cryptography and Lattices Conference*, pages 32–44. Springer, 2001.

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. *E*ndomorphism rings of elliptic curves over finite fields. PhD thesis, University of California at Berkeley, 1996.

Ler22. Antonin Leroux. *Q*uaternion 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 lll-reduction algorithm with quasi-linear time complexity. In *P*roceedings of the forty-third annual ACM symposium on Theory of computing, pages 403–412, 2011.

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.

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. *J*ournal de théorie des nombres de Bordeaux, 7(1):219–254, 1995.

Sho94. P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *P*roceedings 35th Annual Symposium on Foundations of Computer Science, pages 124–134, Nov 1994.

Sim05. Denis Simon. Quadratic equations in dimensions 4, 5 and more. *p*reprint, 2005.

Sut11. Andrew Sutherland. Computing hilbert class polynomials with the chinese remainder theorem. *M*athematics of Computation, 80(273):501–538, 2011.

Sut12. Andrew V Sutherland. Accelerating the cm method. *L*MS Journal of Computation and Mathematics, 15:172–204, 2012.

Sut13. Andrew Sutherland. On the evaluation of modular polynomials. *T*he Open Book Series, 1(1):531–555, 2013.

Vél71. J. Vélu. Isogénies entre courbes elliptiques. *C*omptes-Rendus de l'Académie des Sciences, Série I, 273:238–241, juillet 1971.

Voi18. John Voight. *Q*uaternion Algebras. Springer Graduate Texts in Mathematics series, 2018.

Wat69. William C. Waterhouse. Abelian varieties over finite fields. *A*nnales Scientifiques de l'E.N.S, 1969.

A Proof of Lemma 1

Let us prove Lemma 1.

Proof. To simplify a bit the proof, we do not prove exactly Lemma 1, but a variant where we replace $\Phi(L_i)$ by L_i anytime its appears in the formulation of the proposition. Since we have $L_i = \ell_i^{e_i}$ and $\Phi(L_i) = \ell_i^{e_i-1}(1+\ell_i)$, the proof of Lemma 1 can clearly be derived in a similar manner.

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 take some number $N > B$. Our goal is to construct a *good factor-basis* (as defined in Remark 1).

Let us construct two integers n, Λ_n recursively from $\Lambda_0 = 1$ in the following manner: if $\lambda_i \geq C_1 N$, then set $n = i$, otherwise let $\Lambda_{i+1} = \lambda_{i+1}\Lambda_i$ with $\lambda_{i+1} = \ell_{i+1}^{e_i}$ where ℓ_{i+1} is the $i+1$-th prime and e_i is one if $\ell_i \geq C_0 \log(N)$ or the biggest exponent such that $L_{i+1} < C_0 \log(N)$. It is clear that such our little recursive algorithm always terminates and so we can get two integers n, Λ_n in that manner for any N. Moreover, we have $C_0/2\log(N) \leq \lambda_1$ since $\ell_1 = 2$ and $\sqrt{C_0 \log(N)} \leq \lambda_i$ for all other i. Then, there is two possibilities: either $\Lambda_n < C_2 N$ or $\Lambda_n \geq C_2 N$.

First case: $\Lambda_n < C_2 N$. Then, if we set $L_i = \lambda_{n-i+1}$ for $1 \leq i \leq n$, we will show that $L_1, \ldots, L_n$ is a valid factor basis. By definition of each L_i, it is clear that is suffices to prove that the n-th prime (the divisor of L_1) is smaller than $C_0 \log(N)$. Since $\Lambda_n < C_2 N$, we can take logarithms to get $n/2(\log(C_0) + \log\log(N)) < \Lambda_n < \log(C_2) + \log(N)$ from the bound $\sqrt{C_0 \log(N)}$. It follows that since $N > B$ is "big enough", we can assume $n < 3\log(N)/\log\log(N)$. Similarly, as it is clear than n grows with N, we can assume $\ell_n < 2n\log(n) < 6\log(N)$ by the usual estimates $\ell_n \approx n\log(n)$. Thus, if $C_0 > 6$, we have proven that $L_1, \ldots, L_n$ is a valid factor basis for N.

Second case: $\Lambda_n \geq C_2 N$. Since Λ_n is too big, we will try to remove a factor δ to be able to extract our factor basis from Λ_n/δ. Since N is big enough, we can assume $n \geq 4$. We don't want to touch λ_1 since the bound is tight, but we can remove some powers of ℓ_2, ℓ_3. The idea is that we can remove up to $O(\sqrt{Log(N)}$ from each λ_i. Moreover since $\Lambda_{n-1} < C_1 N$, we can easily show that $\Lambda_n = O(N\log(N))$. Thus, it is clear that we can remove enough from λ_2, λ_3, to get to the point where $\Lambda_N/\delta < C_2 N$. However, we need to preserve the equality $C_1 N \leq \Lambda_N/\delta$. Fortunately, for that it suffices that C_2/C_1 is big enough.

More precisely: since we have $\Lambda_{n-1} < C_1 N$, by the same estimate as before we have $n < 1 + 3\log(N)/\log\log(N)$. Hence, with $\ell_n < 2n\log(n)$ we get $\ell_n < 7\log(N)$. Thus, we can assume $\Lambda_n = \Lambda_{n-1}\lambda_n < 7C_1 N\log(N)$. Thus, we get

$$C_1 N \leq \Lambda < 7NC_1 \log(N). \tag{A.1}$$

Now, we want to find $\delta | \Lambda_n$ such that $C_1 N \leq \Lambda_n/\delta < C_2 N$. We start by using λ_2. Let us take δ_2 as the biggest divisor of λ_2 such that

$$\delta_2 < \min(\Lambda_n/(C_1 N), \sqrt{C_0 \log(N)}/\ell_2).$$

If we assume $C_2 > \ell_2 C_1$, then we have $\delta_2 \geq \ell_2$ and so we can take a non-trivial δ_2. In that case, we get $C_1 N \leq \Lambda_n/\delta_2 < \Lambda_n$.

If $\Lambda_n/\delta_2 < C_2 N$ we are done. Indeed, by definition on δ_2, we now that the lower bound $C_1 N \leq \Lambda_n/\delta_2$ is preserved. Then, since $\delta_2 < \sqrt{C_0 \log(N)}/\ell_2$, we get $\lambda_2/\delta_2 > \lambda_2 \ell_2/\sqrt{C_0 \log(N)}$, and since by definition we have $\lambda_2 > C_0 \log(N)/\ell_2$, we get that $\lambda_2/\delta_2 > \sqrt{C_0 \log(N)}$. Moreover, with the same reasoning as above, if $C_0 > 7$, we have proven that $\lambda_n < C_0 \log(N)$. This prove that setting $L_i = \lambda_{n-i+1}$ for $1 \leq i \neq 2 \leq n$ and $L_{n-1} = \lambda_2/\delta_2$ give us the desired factor basis.

If not and $\Lambda_n/\delta_2 \geq C_2 N$, 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 $\delta_2 > 1/\ell_2 \min(\Lambda_n/(C_1 N), \sqrt{C_0 \log(N)}/\ell_2)$.

If $\Lambda_n/(C_1 N) < \sqrt{C_0 \log(N)}/\ell_2$, then $\delta_2 > \Lambda_n/(\ell_2 C_1 N)$ and so $\Lambda_n/\delta_2 < \ell_2 C_1 N$. This is smaller than $C_2 N$ by our assumption on C_1 and C_2 and so this is a contradiction. Thus we must have $\Lambda_n/(C_1 N) \geq \sqrt{C_0 \log(N)}/\ell_2$ and we can get the lower-bound $\delta_2 \leq \sqrt{C_0 \log(N)}/\ell_2^2$. If we plug that into Eq. (A.1), we obtain the following

$$C_1 N \leq \Lambda/\delta_2 < \frac{7\ell_2^2 C_1}{\sqrt{C_0}} N \sqrt{\log(N)}. \tag{A.2}$$

Now, we define δ_3 as the biggest divisor of λ_3 such that

$$\delta_3 < \min(\Lambda_n/(C_1 \delta_2 N), \sqrt{C_0 \log(N)}/\ell_3).$$

Then, following the same reasoning as for δ_2, we get that if we assume further $C_1 \ell_3 < C_2$, then we can take a non-trivial δ_3. Furthermore, we have that either $\Lambda_n/(\delta_2 \delta_3)$ and we can derive our factor basis, or we must have $\Lambda_n/(\delta_2 C_1 N) \geq \sqrt{C_0 \log(N)}/\ell_3$. In that case, we can plug this in Eq. (A.2) to get

$$C_1 N \leq \frac{\Lambda_n}{\delta_2 \delta_3} < \frac{7\ell_2^2 \ell_3^2 C_1}{C_0} N. \tag{A.3}$$

If we assume that $C_2 > \frac{7\ell_2^2 \ell_3^2 C_1}{C_0}$, then we are also done.

In conclusion, we can derive our log factor-basis for any big enough N assuming that the constants C_0, C_1, C_2 satisfies:

- $C_0 > 7$.
- $C_2 > 3C_1$.
- $C_2 > 5C_1$.
- $C_2 > \frac{1575 C_1}{C_0}$.

(we used $\ell_2 = 3, \ell_3 = 5$). This system can clearly be satisfied and so this proves the result.