COMPUTING MODULAR POLYNOMIALS BY DEFORMATION

SABRINA KUNZWEILER, DAMIEN ROBERT

ABSTRACT. We present an unconditional CRT algorithm to compute the modular polynomial $\Phi_\ell(X,Y)$ in quasi-linear time. The main ingredients of our algorithm are: the embedding of ℓ-isogenies in smooth-degree isogenies in higher dimension, and the computation of m-th order deformations of isogenies. We provide a proof-of-concept implementation of a heuristic version of the algorithm demonstrating the practicality of our approach.

Our algorithm can also be used to compute the reduction of Φ_ℓ modulo p in quasi-linear time (with respect to ℓ) $\tilde{O}(\ell^2(\log p + \log \ell)^{\mathfrak{O}})$.

1. INTRODUCTION

The modular polynomial $\Phi_\ell(X,Y) \in \mathbb{Z}[X,Y]$ parametrizes ℓ-isogenies between elliptic curves. It is a fundamental object for building isogenies and exploring isogeny graphs, and is used in many applications, notably in the Schoof–Elkies–Atkin (SEA) point counting algorithm [40, 17, 1]. We will assume ℓ to be prime for simplicity, the extension to the general case is not hard. It is well known that Φ_ℓ has degree $\ell+1$ in each variable with height $O(\ell \log \ell)$, so the modular polynomial has size $O(\ell^3 \log \ell)$. More explicitly, it is shown in [9] that

$$\log|a_{i,j}| \leq 6\ell\log(\ell) + 16\ell + \min(2\ell, 14\sqrt{\ell}\log\ell), \tag{1}$$

where $\Phi_\ell = \sum_{i,j} a_{i,j} X^i Y^j$. Refined bounds can be found in [6].

1.1. **Algorithms for computing modular polynomials.**

The first quasi-linear algorithm to compute Φ_ℓ is due to Enge in [18] and uses analytic methods. Namely, it uses an evaluation-interpolation approach: fix $\ell+1$ nice period matrices τ_i in the complex upper half plane, and evaluate $\Phi_\ell(j(\tau_i),Y) = \prod_{j=1}^{\ell+1}(Y - j(\tau_{i,j})) = Y^{\ell+1} + \sum_{j=0}^{\ell} c_j(j(\tau_i))Y^j$, where the $\tau_{i,j}$ are period matrices of the $\ell+1$ elliptic curves isogenous to E_{τ_i}. Each coefficient c_j, seen as polynomial in the variable X, can then be recovered by interpolation.

Since the modular polynomial has large height, the analytic method needs to work with large precision. To achieve quasi-linearity, the j-invariants thus need to be evaluated in quasi-linear time in the precision. Due to the use of large precision floating point arithmetic, Enge's algorithm is only heuristic: the assumption is that the loss of precision in the floating point arithmetic does not explode. This has been made rigorous in recent work by Kieffer [25, 23]. In [23], Kieffer focuses on the rigorous evaluation of dimension 2 theta functions; but the same arguments work (and are easier) to compute the modular polynomials in dimension 1.

In practice, the analytic algorithm works well; due to the quasi-linear complexity, the blocking factor to compute large modular polynomials are their size, hence the available memory, rather than the computing power. In [8], Bröker, Lauter and Sutherland introduce a quasi-linear algorithm based on the Chinese remainder theorem (CRT) to compute Φ_ℓ. Besides being purely algebraic, a great advantage is that thanks to the explicit CRT algorithm [4], the reduced modular polynomial $\Phi_\ell(X,Y) \mod m$ (m a large integer) can be computed by using a memory of only $O(\ell^2 \log(m\ell))$ rather than $O(\ell^3 \log \ell)$.

In the CRT algorithm, the modular polynomial $\Phi_\ell(X,Y) \mod p_i$ is computed for many ($O(\ell)$) small primes p_i (with $\log p_i \approx \log \ell$). The Chinese remainder theorem is then used to reconstruct $\Phi_\ell(X,Y) \mod \prod p_i$. To obtain a quasi-linear algorithm, $\Phi_\ell(X,Y) \mod p_i$ needs to be computed in $O(\ell^2 \log^{O(1)} p_i)$. As in the analytic method, this is done by an evaluation-interpolation approach, but in a careful way to achieve the desired complexity.

Fix a small prime $p = p_i \neq \ell$. The naive evaluation-interpolation approach would be to select $\ell+1$ elliptic curves $E_i/\mathbb{F}_p$, and compute all $\ell+1$-isogenous elliptic curves $E_{i,j}/\mathbb{F}_p$. But given an elliptic curve $E_i/\mathbb{F}_p$, it is already not obvious how to compute the $\ell+1$-isogenies efficiently: factoring the division polynomial (of degree ℓ^2) is too expensive. A solution is to require that $E_i[\ell]$ has rational torsion; we can then efficiently sample a basis of it, generate all kernels, and apply the recent square-root Vélu algorithm [5], built on Vélu's formula [46], to compute the isogenous curves. The square-root Vélu algorithm costs $\tilde{O}(\sqrt{\ell})$ arithmetic operations (as opposed to a cost of $O(\ell)$ for the evaluation of the classical Vélu's formula), so computing $\Phi_\ell(j(E_i),Y)$ in this way already costs $\tilde{O}(\ell\sqrt{\ell})$ arithmetic operations. Applying the same algorithm to each E_i would result in $\tilde{O}(\ell^2\sqrt{\ell})$ arithmetic operations and is therefore too expensive for the purpose of constructing a quasi-linear algorithm.

Instead, the solution proposed in [8] is to use isogeny volcanoes to find suitable ℓ-isogenous curves faster than in $O(\ell)$. They use the Hilbert class polynomial of a carefully crafted imaginary quadratic order to ensure that the isogeny volcano has the desired property. Their algorithm computes Φ_ℓ in $O(\ell^3 \log^3 \ell \log\log \ell)$ which is slightly faster than the analytic method, and as mentioned above requires less memory to compute $\Phi_\ell \bmod m$. However, the complexity proof relies on the Generalised Riemann Hypothesis (this is required to be sure that there exist small generators of the class group), hence is not unconditional.

1.2. **New algorithmic tools.** The field of isogeny-based cryptography radically changed following the break of SIDH [11, 31, 39]. It was soon clear that the tools (Kani's lemma, Zahrin's trick) used for the break could also be used to achieve a very efficient representation of isogenies [37] by embedding them into higher dimensional isogenies, which in turn lead to new isogeny-based cryptosystems (for instance SQISignHD [13] and FESTA [3]).

These new algorithmic tools are not restricted to isogeny-based cryptography; in [38] the second author uses them to outline new algorithms to compute the endomorphism ring and canonical lifts of ordinary abelian varieties in polynomial time. A sketch of a quasi-linear algorithm to compute modular polynomials is described as well.

We summarize the efficient representation of isogenies of [37], which we will call an *HD representation* (for higher dimensional representation), as follows:

Theorem 1.1. *Let $f : E_1 \to E_2$ be an ℓ-isogeny between elliptic curves over a finite field $\mathbb{F}_q$. There exists an HD representation F of f which can be used to* *evaluate $f(P)$ for $P \in E_1(\mathbb{F}_q)$ in time $\tilde{O}(\log^{\mathfrak{O}} \ell)$ arithmetic operations in $\mathbb{F}_q$, for some constant $\mathfrak{O}$ (not depending on q or ℓ)*[1].

If E_1 has full rational 2^n-torsion with $2^n > \ell$, there exists an HD-representation for which the evaluation takes $O(\log \ell)$ arithmetic operations.

An HD-representation with $\mathfrak{O} = 1$, as in the last part of the theorem, is called *special HD representation.* The key point of the SIDH attacks is that an elliptic curve isogeny f can be embedded into a higher dimensional N-isogeny as long as we know the evaluation of f on a basis of $E_1[N]$, $N > \ell$. The first representation in Theorem 1.1 follows by taking N powersmooth, and the second (which more generally could be used as long as the 2^n-torsion lies in a small extension of $\mathbb{F}_q$) by taking $N = 2^n$. Of course, we also have a special HD representation if the $3^n > \ell$-torsion is rational and so on. We refer to [37, 38] for more details.

1.3. **Our contributions.** The goal of the current article is to give a fully fledged and rigorous analysis of the algorithm to compute modular polynomials sketched in [38, Section 6]. Notably, we prove in Section 5 the following theorem, which matches the complexity of [8].

Theorem 1.2. *There exists an unconditional CRT algorithm which computes the modular polynomial $\Phi_\ell(X, Y)$ in quasi-linear time $O(\ell^3 \log^3 \ell \log\log \ell)$.*

More precisely, we present a rigorous algorithm, along with a faster but heuristic version, which is the one we implemented. We remark that this can also be used to compute the reduction modulo m of Φ_ℓ in quasi-linear space $O(\ell^2 \log(m\ell))$. To simplify the exposition, we will always assume that $\ell > 2$ and the base characteristic $p > 2$, too.

We briefly explain the main ideas behind the algorithms. The key tool is the following theorem proved in Section 3, and stated here in an informal way:

Theorem 1.3. *Given an HD representation of an ℓ-isogeny $f : E_1 \to E_2$ over a finite field $\mathbb{F}_q = \mathbb{F}_{p^d}$, with ℓ prime to the characteristic p, and given an m-th order deformation of E_1 to an elliptic curve $\mathcal{E}_1/R$ with $R = \mathbb{F}_q[\epsilon]/(\epsilon^{m+1})$, then we can efficiently compute the deformation of f to $\mathcal{E}_1$, that is, $\mathcal{E}_2$ an elliptic curve over R and an isogeny $\tilde{f} : \mathcal{E}_1 \to \mathcal{E}_2$ with $\tilde{f} \equiv f \pmod{\epsilon}$ in $\tilde{O}(\log^{\mathfrak{O}} \ell)$ arithmetic operations in R.*

If we have a special HD representation of f, then we can compute $\tilde{f}$ in $O(\log \ell)$ arithmetic operations in R.

In other words: once we have an efficient representation of an isogeny, we can efficiently deform it. Deformation techniques are not new in computer algebra, see for instance [41]. Here we exploit them in order to recover the modular polynomial Φ_ℓ modulo a prime p:

Corollary 1.4. *Let $\mathbb{F}_q$ be a field with char$(\mathbb{F}_q) = p$ and $E_0/\mathbb{F}_q$ be an elliptic curve which has all its $\ell + 1$ ℓ-isogenies rational. Given an HD representation of these isogenies, we can compute $\Phi_\ell \mod p$ in time $\tilde{O}(\ell^2 \log q)$. If we have a special HD representation of all isogenies, the cost is $O(\ell^2 \log^2 \ell)$ operations over $\mathbb{F}_q$.*

Proof. The idea is to use the HD representation of the isogenies to deform them to $\mathbb{F}_q[\epsilon]/(\epsilon^{m+1})$. When the precision m is high enough, we get enough information to reconstruct the full modular polynomial. We refer to Section 4 for more details. □

[1]In practice, we can achieve $\mathfrak{O} = 12$ using [38, Proposition 2.9].

We remark that if $q = p^d$ with d small, then Corollary 1.4 is quasi-linear in its output size $O(\ell^2 \log p)$. To apply Corollary 1.4, we need to find an elliptic curve over a small extension of $\mathbb{F}_p$ on which we can efficiently compute the $\ell + 1$ ℓ-isogenies.

The easiest case is when we can find an elliptic curve $E_0/\mathbb{F}_q$ with full rational ℓ-torsion. In this case we compute a basis, compute all kernels, and apply Vélu's formula. Given a basis of $E_0[\ell]$, computing all the ℓ-isogenies costs $O(\ell^2)$ arithmetic operations. [2]

For instance, if $p = 3 \pmod 4$ and $\ell \mid p^2 - 1$, then $E : y^2 = x^3 + x$ is a supersingular curve over $\mathbb{F}_p$ and $E(\mathbb{F}_{p^4}) = (\mathbb{Z}/(p^2-1)\mathbb{Z})^2$, then we can efficiently sample a basis of the ℓ-torsion in $O(\log p)$ operations in $\mathbb{F}_{p^4}$ (in practice we can work with $E/\mathbb{F}_{p^2}$ or its quadratic twist according to whether $\ell \mid p+1$ or $\ell \mid p-1$). For technical reasons (to kill the extra automorphisms of $y^2 = x^3 + x$), we will rather work with a 2-isogenous curve , that is we take $E_0 : y^2 = x^3 + 6x^2 + x$.

If we add the congruence condition that $2^n \mid p^2 - 1$ for some $2^n > \ell$, so that the 2^n-torsion of E_0 is rational over $\mathbb{F}_{p^4}$, then we can even use a special HD representation of the ℓ-isogenies.

Proposition 1.5. *For $p \in \mathcal{P}_\ell^* = \{p > 11$ prime : $2^n \cdot \ell \mid p+1$, where $n = \lceil \log_2(\ell) \rceil\}$, we can compute $\Phi_\ell \mod p$ in $O(\log p + \ell^2)$ arithmetic $\mathbb{F}_p$-operations.*

We prove this proposition in Section 4.4. We call a prime $p \in \mathcal{P}_\ell^*$ a suitable CRT prime. By Dirichlet's theorem on primes in arithmetic progressions, we know that there are sufficiently many such primes, with the appropriate density (see Lemma 4.3). Using these suitable CRT primes, we obtain Theorem 1.2.

We remark that the idea to use supersingular curves for modular polynomial computation is not new, and in [29] Leroux also gives a heuristic algorithm relying on supersingular isogenies to compute modular polynomials modulo a prime.

In practice, in the implementation, we use a subset $\mathcal{P}_\ell \subset \mathcal{P}_\ell^*$, defined in Section 4.1, where we replace the condition $n = \lceil \log_2(\ell) \rceil$ (which is enough to ensure that $2^n - \ell$ can be written as a sum of four squares) by a condition $2^n - c_\ell \ell = a^2 + 4b^2$. When $p \in \mathcal{P}_\ell$, this stronger condition ensures that we can find a special HD representation in dimension 2 of our isogenies rather than in dimension 8. We prove this version of Proposition 1.5 for $p \in \mathcal{P}_\ell$ in Section 4.2. While this does not change the asymptotic complexity of the algorithm, it improves the constant and greatly simplifies the implementation. However, to achieve the quasi-linear CRT algorithm of Theorem 1.2 when using primes in $\mathcal{P}_\ell$, we need to rely on a heuristic (Heuristic 4.1), which says that we can find a suitable n such that 2^n is not too large compared to ℓ. In this paper, we will give detailed algorithms for Theorems 1.2 and 1.3, Corollary 1.4, and Proposition 1.5 for the special case when $p \in \mathcal{P}_\ell$, because it is the one we implemented, and briefly explain how to generalize to the general cases.

1.4. Generalisations and perspectives.

The techniques used in our algorithms, can also be applied to other settings. Here, we highlight some generalizations.

Computing Φ_ℓ modulo an arbitary prime p. In this paper, we explain the computation for primes $p \in \mathcal{P}_\ell^*$. This may be extended to general primes p as follows. We also use supersingular curves, because all their ℓ-isogenies are rational over $\mathbb{F}_{p^2}$. We first need to sample a supersingular elliptic $E_0/\mathbb{F}_{p^2}$ with explicit

[2]This is not the dominating step of the algorithm, hence there is no need to apply the asymptotically faster square-root Vélu algorithm.

known endomorphism ring and endomorphism action. This can be seen as a precomputation depending only on p and not on ℓ. For instance, we can apply Bröker's algorithm [7], which is in $O(\log^3 p)$ under GRH.

The explicit endomorphism ring action of E_0 allows us to compute the $\ell + 1$ ℓ-isogenies in polynomial time in $\log \ell + \log p$ by [47] under GRH, or unconditionally using the more recent CLAPOTIS method [35, Remark 2.10]. Namely, we compute the ideals in $End(E_0)$ corresponding to the $\ell + 1$ ℓ-isogenies, and then we use CLAPOTIS to compute an HD representation of the ℓ-isogenies associated to these ideals in $\tilde{O}((\log \ell + \log p)^{\mathfrak{O}})$ arithmetic operations by ideal.

This initialisation is only interesting when $\log p$ is sufficiently small compared to ℓ; it is thus not suitable in applications like point counting where we usually have $\log p \approx \ell$. (Of course conversely if $\log p$ is sufficiently large compared to ℓ, it is faster to just compute Φ_ℓ directly.) An alternative initialisation approach is described in [38, 6.2] with better complexity in $\log p$, and another heuristic approach is described in [29].

From the discussion above, we get:

Corollary 1.6. *Let p be a fixed prime. Assume that we are given an explicit supersingular curve $E_0/\mathbb{F}_{p^2}$ with known full endomorphism ring (as mentionned above such an E_0 can be computed in $\tilde{O}(\log^3 p)$ under GRH). Then there is a quasi-linear algorithm to compute $\Phi_\ell \mod p$ in $O(\ell^2(\log p + \log \ell)^{\mathfrak{O}})$, for some constant $\mathfrak{O}$ which does not depend on ℓ or p.*

In particular, under GRH, we obtain an algorithm that can compute $\Phi_\ell \mod p$ in $\tilde{O}_p(\ell^2)$, where the notation $\tilde{O}_p$ means that the constants depend on p.

p-adic lifting. One can extend Theorem 1.3 to more general Artinian rings with residue field $\mathbb{F}_q$, notably rings of the form $R = \mathbb{Z}_q[\epsilon]/(\epsilon^{m_1}, p^{m_2})$. This allows us to combine our horizontal deformation with a p-adic lifting, and give a p-adic lifting version of Corollary 1.4 which results in an alternative way to compute the modular polynomial Φ_ℓ in quasi-linear time. However, the CRT algorithm is better because it can be used to compute $\Phi_\ell \mod m$ by the explicit CRT.

Modular polynomials in higher dimension. In this paper, for simplicity we confine ourselves to computing deformations of isogenies of elliptic curves. Using theta models for abelian varieties, our methods to compute these deformations efficiently (Theorem 1.3) extend naturally to higher dimension. In subsequent work, we will explain how this paves the way to computing higher dimensional Siegel modular polynomials in quasi-linear time. Such a complexity was conjectured in [36, Conjecture 5.3.14]. The reason we focus on the dimension 1 case in this current paper is by lack of space, for the simplicity of the exposition[3], and also because we have only implemented this case so far.

For modular polynomials in higher dimension, there is currently no known quasi-linear analytic method. An important step in this direction is achieved in a recent work by Elkies and Kieffer, who developed a quasi-linear algorithm to evaluate theta functions in quasi-linear time in the precision in all dimensions. However, to

[3]The main difficulty in higher dimension is that we now need to reconstruct multivariate rational functions from power series rather than simply polynomials; and also take into account the algebraic equations between the modular invariants.

compute the modular polynomial efficiently, one would also need a quasi-linear algorithm for computing period matrices from theta constants, because the evaluation-interpolation approach requires to select suitable points of interpolation to obtain a fast multivariate interpolation algorithm. Such an algorithm is only known in dimensions 1 and 2. The dimension 2 case follows from heuristic work of Dupont [16], and was then rigorously proven by Kieffer in [24]. Kieffer thus describes in [23] a rigorous algorithm to evaluate modular polynomials in dimension 2, which combined with [16, 32] gives a rigorous analytic algorithm in dimension 2 to compute the full modular polynomial.

Likewise, the CRT approach of [8] crucially uses the volcano structure; but isogeny graphs of principally polarised abelian varieties over a finite field are more complex, so it is not obvious how to extend this approach to higher dimensions.

By contrast, as we have mentioned, our deformation method generalises in a natural way to higher dimensions. We expect that similar deformation techniques could also be used to compute more general modular correspondences or actions of Hecke operators on modular forms.

1.5. **Outline.** Section 2 contains theoretical background on deformations. The results of that section are made explicit in Section 3, where we describe different methods for computing with deformations. In Section 4, we present an explicit algorithm to compute modular polynomials over finite fields. Building on this, we describe a CRT algorithm for computing the modular polynomial in Section 5.

2. Deformations and isogenies

In this section, we provide an overview on the topic of deformations of principally polarised abelian varieties, and in particular we discuss deformations of isogenies. For more detailed explanations, the reader is referred to [19, 42]. Throughout, k denotes a field with characteristic $\mathrm{char}(k) = p > 2$, and R is an Artinian ring of the form $R = k[\epsilon]/(\epsilon^{m+1})$ for some integer $m \geq 0$.[4]

2.1. **Deformations of (principally polarised) abelian varieties.** Recall that an abelian variety is a group scheme over a field k which is also a proper, geometrically integral variety over k. This fits into the following more general framework of abelian schemes.

Definition 2.1. An *abelian scheme* over a scheme S is a group scheme $\mathcal{A} \to S$ which is smooth, proper and has geometrically connected fibres. If $S = \mathrm{Spec}(k)$, then we call $\mathcal{A}$ an abelian variety.

In our applications, we always have $S = \mathrm{Spec}(R)$ with $R = k[\epsilon]/(\epsilon^{m+1})$ as above, but everything extends to a general local Artinian ring R with maximal ideal m_R and residue field $R/m_R = k$. Note that in this setting, along with the map $\mathrm{Spec}(k) \to \mathrm{Spec}(R)$ induced by the projection $R \to R/(\epsilon)$, there is also a map $\mathrm{Spec}(R) \to \mathrm{Spec}(k)$ induced by the canonical inclusion $k \hookrightarrow R$.

Example 2.2. Let $\mathcal{E} \to \mathrm{Spec}(R)$ be an abelian scheme of dimension 1, in other words an elliptic curve over R. As is the case for elliptic curves over fields, we

[4]The exponent $m + 1$ is chosen, since we work with m-th order deformations (Definition 2.3).

can make Definition 2.1 more explicit in this situation. Since R is local, it can be represented as a subscheme of $\mathbb{P}^2_R$ defined by an equation of the form

$$\mathcal{E} : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

with $a_1, a_2, a_3, a_4, a_6 \in R$. Note that $\operatorname{Spec}(R) = \{(\epsilon)\}$. Therefore, the scheme $\mathcal{E}$ has only one fibre $E = \mathcal{E}_{(\epsilon)}$. It is obtained by reduction modulo (ϵ).

Note that points of $\mathcal{E}$ are sections $s : \operatorname{Spec}(R) \to \mathcal{E}$. More explicitly, we may also view a point $P \in \mathcal{E}(R)$ as usual by three projective coordinates, i.e. $P = (x : y : z) \in \mathcal{E} \subset \mathbb{P}^2_R$.

Definition 2.3. Let A be an abelian variety over k. We say that an abelian scheme $\mathcal{A} \to \operatorname{Spec}(R)$ is an *m-th order deformation* of A if its special fibre $\mathcal{A} \times_R k \to \operatorname{Spec}(k)$ is isomorphic to A.

In our setting, along the special fibre map $\operatorname{Spec}(k) \to \operatorname{Spec}(R)$ induced by $R \to R/(\epsilon)$, there is also a map $\operatorname{Spec}(R) \to \operatorname{Spec}(k)$ induced by the canonical inclusion of k into R. For any abelian variety A, there is a trivial deformation given by $\mathcal{A} = A \times_k \operatorname{Spec}(R)$. In general, the group structure of a deformation $\mathcal{A}$ of A is not obvious. However, we can describe the étale part of the torsion group explicitly which is explained in the next remark.

Remark 2.4. Let N be an integer coprime to p, and consider a deformation $\mathcal{A} \to \operatorname{Spec}(R)$ of an abelian variety A. Since $\mathcal{A}[N]/\operatorname{Spec}(R)$ is étale and R is Henselian, we have a canonical isomorphism $\mathcal{A}[N](R) \cong A[N](k)$ (see [43, Tag 04GG]). This isomorphism will be made explicit in Algorithm 2 for elliptic curves. In fact, because R is Henselian, the functor which associates to a finite étale cover $\mathcal{X}/\operatorname{Spec}(R)$ its special fibre $X/\operatorname{Spec}(k)$ is an equivalence of categories between finite étale covers of $\operatorname{Spec}(R)$ and finite étale covers of $\operatorname{Spec}(k)$ (see [43, Tag 0A48]). Thus $A[N]$ deforms uniquely to $\mathcal{A}[N]/\operatorname{Spec}(R)$, in our setting this is simply given by the trivial deformation.

We remark that in our applications, we always work with points of $\mathcal{A}[N]$, and only compute separable isogenies. In these cases, we can use the standard addition laws and formulae for isogeny computation known for abelian varieties over fields. Indeed, in [33, § 6], Mumford construct the universal abelian scheme of level n over $\mathbb{Z}[1/n]$ via Riemann's relations, in level divisible by 8 (the construction was then extended by Kempf to level divisible by 4). In particular, the addition law and ℓ-isogeny formulas which use these Riemann relations have good reduction over $\mathbb{Z}[\frac{1}{2\ell}]$, hence are valid over R as long as $p \neq 2, \ell$.

Although we won't need it, we can make explicit the full structure of $\mathcal{A}(R)$ as follows. The kernel of the reduction map $\mathcal{A}(R) \to A(k)$ is given by $\mathcal{A}(p)^0(R)$, where $\mathcal{A}(p)^0$ is the connected component of the p-divisible group of $\mathcal{A}$. We have $\mathcal{A}(p)^0(R) = \tilde{\Gamma}(m_R)$ where $\tilde{\Gamma}$ is the formal Lie group associated to $\mathcal{A}$. Furthermore, since k is perfect, the connected-étale sequence splits over k, hence we have an exact sequence $0 \to \mathcal{A}(p)^0(R) = \tilde{\Gamma}(m_R) \to \mathcal{A}(p)(R) \to \mathcal{A}(p)_{\text{étale}}(R) = A(p)_{\text{étale}}(k) \to 0$. We refer to [44] for more details.

Note that, as proved by Grothendieck (see [34, §2.2]), the deformation space of a g-dimensional abelian variety has dimension g^2 which is equal to the dimension of the tangent space. We now specialize to the case of principally polarised abelian varieties. Recall that a *polarisation* L on an abelian variety A is given by an ample line bundle (defined up to translation and potentially over a separable field extension of k) which defines a rational morphism to the dual of A, $\phi_L : A \to \hat{A}$. And it is called *principal* if ϕ_L is an isomorphism.

Definition 2.5. Let (A, L) be a principally polarised abelian variety over k. We say that $(\mathcal{A}, \mathcal{L})$ is an *m-th order deformation* of (A, L), if $\mathcal{A}$ is a deformation of A and $\mathcal{L}$ is an ample line bundle on $\mathcal{A}$ with $\mathcal{L} \times_R \mathrm{Spec}(k) \cong L$.[5]

If the polarisation is clear from the context, we write A (resp. $\mathcal{A}$) instead of (A, L) (resp. $(\mathcal{A}, \mathcal{L})$). The deformation space of principally polarised abelian varieties has dimension $g(g + 1)/2$. More precisely, Grothendieck and Mumford proved that for a principally polarised abelian variety A/k, the functor of deformations of A is pro-representable by $k[[t_1, \ldots, t_{g(g+1)/2}]]$ (see [34, §2.3]). Concretely, it is given by the completion at A of the moduli space A_g/k of principally polarised abelian varieties[6]. Essentially, this means that any m-th order deformation $\mathcal{A}$ of A corresponds to a unique ring homomorphism

$$\psi_{\mathcal{A}} : k[[t_1, \ldots, t_{g(g+1)/2}]] \to R. \tag{2}$$

This motivates the following definition.

Definition 2.6. Let A be a principally polarised abelian variety over k, and $\mathcal{A}$ an m-th order deformation of A. Then we say that

$$\lambda_1 = \psi_{\mathcal{A}}(t_1), \ldots, \lambda_{g(g+1)/2} = \psi_{\mathcal{A}}(t_{g(g+1)/2})$$

with ψ_A as in Eq. 2 are *deformation parameters* of $\mathcal{A}$.

Note that the definition of the deformation parameters depends on the local representation $k[[t_1, \ldots, t_{g(g+1)/2}]]$ of the moduli space at A. Below, we describe a canonical choice for the case $g = 1$.

Example 2.7. In the case of elliptic curves, the deformation space is one-dimensional, hence a deformation is defined by a single deformation parameter λ.

Let E be an elliptic curve and suppose that $j(E) \neq 0, 1728$. For an m-th order deformation $\mathcal{E}$ of E, we can define the deformation parameter

$$\lambda_{\mathcal{E}} = j(\mathcal{E}) - j(E) \in (\epsilon) \triangleleft k[\epsilon]/(\epsilon^{m+1})$$

associated to the j-invariant.

Note that as long as $j(E) \neq 0, 1728$, we can construct the universal elliptic curve over the universal deformation ring. More explicitly, for a given deformation parameter $\lambda \in (\epsilon)$, we can consider

$$\mathcal{E} : y^2 = x^3 + ax + b, \quad \text{where } a = b = \frac{27(j(E) + \lambda)}{4(1728 - (j(E) + \lambda))},$$

which defines an elliptic curve over R with j-invariant $j(\mathcal{E}) = j(E) + \lambda$. Over $j(E) = 0, 1728$ the universal elliptic curve only exists as an algebraic stack [15], which is why we are going to use $y^2 = x^3 + 6x^2 + x$ rather than $y^2 = x^3 + x$ as the initial point in our algorithms.

[5]A slight technicality: deforming the polarisation means deforming the isogeny ϕ_L. However, if ϕ_L is induced by a line bundle over k (rather than over its separable closure), and ϕ_L deforms to R, then L does too. This is because the obstruction to descending L is given by a smooth torsor, which has a point over k by assumption, so over R too since R is Henselian.

[6]To handle deformations to a general Artinian ring, we would need to use the completion $\hat{\mathsf{A}}_{g,A}/\mathbb{Z} \simeq \mathbb{Z}_q[[t_1, \ldots, t_{g(g+1)/2}]]$ at A of the moduli space over $\mathbb{Z}$ rather than over k.

2.2. **Deformations of isogenies.** Let $f : A \to A'$ be an isogeny of principally polarized abelian varieties over a field k. Here, we discuss deformations of such an isogeny to R. In particular, we are interested in deformations of product isogenies, i.e. isogenies between decomposable principally polarized abelian varieties, and the behaviour of deformation parameters under isogenies.

Recall that a morphism of group schemes $f : \mathcal{A} \to \mathcal{A}'$ over $\mathrm{Spec}(R)$ is an isogeny if it is surjective and its kernel is a flat finite $\mathrm{Spec}(R)$-group scheme.

Proposition 2.8. *Let $f : A \to A'$ be a separable isogeny of principally polarised abelian varieties defined over k. Consider an m-th order deformation $\mathcal{A}$ of A. Then up to isomorphism there exists a unique deformation $\mathcal{A}'$ of A' so that f lifts to an isogeny $\tilde{f} : \mathcal{A} \to \mathcal{A}'$ over R.*

This is a classical result, which stems from the fact that the forgetful map $\mathsf{A}_g(\Gamma_0(\ell)) \to \mathsf{A}_g$ is étale. Recall that $\mathsf{A}_g(\Gamma_0(\ell))$ denotes the moduli space of principally polarized abelian varieties with $\Gamma_0(\ell)$-structure, i.e. the elements are principally polarized abelian varieties together with the kernel of an ℓ-isogeny. Since we need to lift isogenies explicitly in our applications, we provide a short constructive proof below.

Proof of Proposition 2.8. Let G be the kernel of the isogeny $f : A \to A'$. By assumption, f is separable, hence $G \subset A[N]$ where N is coprime to p. Now consider the deformation $\mathcal{A}$ of A to R. Since R is Henselian, it has the same finite étale covers as k. It follows that there is a unique lift $\tilde{G} \subset \mathcal{A}[N]$ of G. We can set $\mathcal{A}' = \mathcal{A}/\tilde{G}$ which is again a group scheme. Necessarily $\mathcal{A}'$ is a deformation of A', and the morphism $\tilde{f} : \mathcal{A} \to \mathcal{A}/\tilde{G}$ is an isogeny lying above f. □

The following proposition tells us how deformations behave under isogenies. To obtain a description in terms of deformation parameters, we assume that some canonical choice on the local representation of the moduli space has been made. We simply refer to *the* deformation parameters, when we mean the deformation parameters induced by this choice.

Proposition 2.9. *Let $f : A \to A'$ be a separable isogeny of principally polarized abelian varieties over k and $m \geq 1$, then there exist polynomials*

$$h_1, \dots, h_{g(g+1)/2} \in k[x_1, \dots, x_{g(g+1)/2}]$$

of degree at most m with the following property:
For any m-th order deformation $\mathcal{A}$ of A with parameters $(\lambda_1, \dots, \lambda_{g(g+1)/2})$, the deformation $\mathcal{A}'$ with parameters $(\lambda'_1, \dots, \lambda'_{g(g+1)/2})$, where

$$\lambda'_i = h_i(\lambda_1, \dots, \lambda_{g(g+1)/2})$$

is the unique deformation of A' for which f lifts to an isogeny $\tilde{f} : \mathcal{A} \to \mathcal{A}'$.

Proof. Let $\hat{\mathsf{A}}_{g,A} = k[[t_1, \dots, t_{g(g+1)/2}]]$ and $\hat{\mathsf{A}}_{g,A'} = k[[t'_1, \dots, t'_{g(g+1)/2}]]$ be the local completions of the moduli space A_g at A and A', that induce our canonical choice of deformation parameters. The isogeny f corresponds to a point $(A, \mathrm{Ker}\, f) \in \mathsf{A}_g(\Gamma_0(\ell))$. The modular correspondence $\overline{\Phi_\ell} : \mathsf{A}_g(\Gamma_0(\ell)) \to \mathsf{A}_g \times \mathsf{A}_g$ is given by two étale projection maps. This induces an (analytic, i.e. continuous) isomorphism of the completion of $\mathsf{A}_g(\Gamma_0(\ell))$ at $(A, \operatorname{Ker} f)$ with $\hat{\mathsf{A}}_{g,A}$ and $\hat{\mathsf{A}}_{g,A'}$ respectively. In particular, f induces an isomorphism $\phi_f : \hat{\mathsf{A}}_{g,A} \to \hat{\mathsf{A}}_{g,A'}$. Thus, we can write

$$t'_i = \tilde{h}_i \in k[[\phi_f(t_1), \dots, \phi_f(t_{g(g+1)/2})]]$$

for each $i \in \{1, \dots, g(g+1)/2\}$. Now the polynomials $h_1, \dots, h_{g(g+1)/2}$ from the statement of the proposition are obtained by truncating these formal power series at degree m. □

Remark 2.10. The modular polynomial Φ_ℓ describes the image of the modular correspondence $\overline{\Phi_\ell} : \mathsf{A}_1(\Gamma_0(\ell))/k \to \mathsf{A}_1/k \times \mathsf{A}_1/k$, it can also be seen as $\overline{\Phi_\ell}$ evaluated on the generic point of A_1/k. From this point of view Corollary 1.4 is natural: we start with the modular correspondence $\overline{\Phi_\ell}$ evaluated at one point $E \in \mathsf{A}_1/k$, i.e. with all ℓ-isogenies starting from E. More precisely, the fibre $(\pi_1 \circ \overline{\Phi_\ell})^{-1}(E)$ gives the isogenies and evaluating $(\pi_2 \circ \overline{\Phi_\ell})$ on the fibre gives the codomains. We then evaluate $\overline{\Phi_\ell}$ at the universal deformation of E at a high enough precision to recover $\overline{\Phi_\ell}$ on the generic point.

2.3. **Deformations of product isogenies.**

We now restrict ourselves to the special case of product isogenies in dimension 2, that is isogenies between decomposable principally polarized abelian surfaces. Let us first recall Kani's Lemma which explains the relation between isogeny diamonds and product isogenies.

Definition 2.11. Let E, E_a, E_b, E_{ab} be elliptic curves and f, g, f', g' isogenies of degree $d_a = \deg(f) = \deg(f')$ and $d_b = \deg(g) = \deg(g')$ that fit into the following commutative diagram.

(3)
$$\begin{tikzcd} E \arrow[r, "f"] \arrow[d, "g"] & E_a \arrow[d, "g'"] \\ E_b \arrow[r, "f'"] & E_{ab} \end{tikzcd}$$

Then $g \circ f' = f \circ g'$ is called a (d_a, d_b)*-isogeny diamond.*

Lemma 2.12 (Kani's Lemma [22])**.** *Let $g \circ f' = f \circ g'$ be a (d_a, d_b)-isogeny diamond and denote $N = d_a + d_b$. Then $F = \begin{pmatrix} f & \hat{g}' \\ -g & \hat{f}' \end{pmatrix}$ is an N-isogeny $F : E \times E_{ab} \to E_a \times E_b$. If $\gcd(d_a, d_b) = 1$, then*

$$\ker(F) = \langle (-\hat{g}(P_N), f'(P_N)), (-\hat{g}(Q_N), f'(Q_N)) \rangle,$$

where $E_a[N] = \langle P_N, Q_N \rangle$.

Note that Kani's Lemma can be generalised to abelian varieties of arbitrary dimension, see [39], and is the key to the HD representation of isogenies in [37].

The following corollary is an easy consequence of the unique lifting of isogenies (Proposition 2.8) and Kani's lemma. It describes when product isogenies lift to product isogenies.

Corollary 2.13. *Let E, E_{ab} be elliptic curves over k and $F : E \times E_{ab} \to E_a \times E_b$ a product isogeny. Consider an m-th order deformation $\mathcal{E}$ of E. Then up to isomorphism there exist unique deformations $\mathcal{E}_{ab}$, $\mathcal{E}_a$, $\mathcal{E}_b$ of E_{ab}, E_a, E_b, so that F lifts to a product isogeny*

$$\tilde{F} : \mathcal{E} \times \mathcal{E}_{ab} \to \mathcal{E}_a \times \mathcal{E}_b.$$

Remark 2.14. Let $F : E \times E_{ab} \to E_a \times E_b$ as in Corollary 2.13. Further let $\mathcal{E}$ and $\mathcal{E}_{ab}$ be arbitrary deformations of E and E_{ab}. Then the isogeny F lifts uniquely to an isogeny $\tilde{F} : \mathcal{E} \times \mathcal{E}_{ab} \to \mathcal{A}$, where $\mathcal{A}$ is a deformation of $E_a \times E_b$. To understand, when $\tilde{F}$ is a product isogeny, we recall from Kani's lemma that any such isogeny is induced by an isogeny diamond as in Eq. 3. This means that the deformation $\mathcal{A}$ is a product if and only if there is an isogeny $\tilde{h} : \mathcal{E} \to \mathcal{E}_{ab}$ lifting the isogeny $f' \circ g = g' \circ f$ underlying the product isogeny F.

The next result describes a convenient tool for computing deformations of product isogenies. This makes use of the Siegel modular cusp form χ_{10}. Recall that

$$\chi_{10}(\mathrm{Jac}(C)) = -2^{-12} \cdot \mathrm{disc}(f), \qquad \chi_{10}(E \times E') = 0,$$

where $C : y^2 = f(x)$ is a genus-2 curve, and E, E' are elliptic curves. In particular χ_{10} can be used to distinguish decomposable from indecomposable elements in the moduli space.

The following corollary is a consequence of Proposition 2.9 applied to product isogenies. To make a precise statement, we require that isogenies are *normalised*. Essentially, this means that the isogeny acts as the identity on the basis of differentials.

Corollary 2.15. *Let E, E_{ab} be elliptic curves over a field k and $F : E \times E_{ab} \to E_a \times E_b$ a product isogeny. Then there exists a polynomial $h \in k[x_1, x_2]$ of degree at most m with the following property:*

Let $\mathcal{E}$ and $\mathcal{E}_{ab}$ be arbitrary m-th order deformations of E and E_{ab} with deformation parameters $\lambda = j(\mathcal{E}) - j(E)$ and $\lambda_{ab} = j(\mathcal{E}_{ab}) - j(E_{ab})$, respectively. Further let $\tilde{F} : \mathcal{E} \times \mathcal{E}_{ab} \to \mathcal{A}$ be a lift of F and assume the isogeny is normalised. *Then*

$$\chi_{10}(\mathcal{A}) = h(\lambda, \lambda_{ab}).$$

Proof. On a decomposable abelian surface $A = E \times E_{ab}$, a canonical choice of deformation parameters is given by choosing two deformation parameters λ and λ_{ab} for E and E_{ab}, respectively. Note that altering only these two deformation parameters preserves the product structure on the resulting deformation. Let $S \subset \hat{\mathsf{A}}_{2,E \times E_{ab}}$ be the subring of the deformation ring given by product deformations.

A difference with Proposition 2.9 is that χ_{10} is a modular form rather than a modular function. Let $\chi_{10,\ell}$ be the modular form which associates to a normalised ℓ-isogeny $F : (A, \omega_A) \to (B, \omega_B)$ the value $\chi_{10}(B, \omega_B)$. (Normalised means that $F^*\omega_B = \omega_A$.) This is a modular form of level $\Gamma_0(\ell)$, hence a section of a power of the Hodge line bundle $\mathfrak{h}^{10}$ on $\mathsf{A}_2(\Gamma_0(\ell))$. [7]

Now given curve equations for $\mathcal{E} \times \mathcal{E}_{ab}/R$, we get a canonical basis of differentials $((dx/y, 0), (0, dx_{ab}/y_{ab}))$, hence a trivialisation of the Hodge bundle $\mathfrak{h}$ above S, i.e. an isomorphism $S \to S \otimes_{\mathsf{A}_2} \mathfrak{h}$. We can use the modular correspondence $\overline{\Phi_\ell}$ as in Proposition 2.9 to see S as a deformation ring of the isogeny F, and pulling back the trivialisation of the Hodge line bundle defined above, we can interpret $\chi_{10,\ell} \in \Gamma(S \otimes_{\mathsf{A}_2} \mathfrak{h}^{10})$ as an element of S, given by a polynomial h. □

Remark 2.16. In Section 3, we will apply Corollary 2.15 to a $(2^n, 2^n)$-isogeny F induced by an isogeny diamond. The isogeny computation consists of a gluing isogeny, followed by a chain of Richelot isogenies, followed by a splitting isogeny,

[7]Note that over $\mathbb{C}$, $\chi_{10,\ell}$ corresponds to the modular forms of level $\Gamma_0(\ell)$: $\tau \mapsto \chi_{10}(\tau/\ell)$.

hence we need to ensure that each step is normalised. The differentials are implicitly kept track off by the curve equation: to an elliptic curve $y^2 = x^3 + ax + b$ we associate the differential dx/y, and to an hyperelliptic curve of genus 2 the differentials $(dx/y, xdx/y)$. Richelot formulas work at the level of curve equations and give normalised formulas.

3. Computing with deformations

In this section we explain different algorithms related to the computation of deformations. This includes the lifting of torsion points to deformed elliptic curves (Algorithm 2), as well as the lifting of $(2, 2)$-isogeny chains (Algorithm 5). Further, we present a method for lifting product isogenies (Algorithm 4) which lies at the heart of our algorithms for the computation of modular polynomials. The section concludes with the deformation of general isogenies and proves Theorem 1.3.

As in the previous section, k is a field with characteristic $p \neq 0$, and $R = k[\epsilon]/(\epsilon^{m+1})$.

3.1. **Arithmetic in $R = \mathbb{F}_q[\epsilon]/(\epsilon^{m+1})$.**

In practice, we work with a finite field $k = \mathbb{F}_q$ in all algorithms. Note that computing in the Artin ring R is equivalent to computing in the formal power series ring $\mathbb{F}_q[[\epsilon]]$ with precision $m + 1$.

Throughout, we use the Schönhage-Strassen bound

$$\mathsf{M}(n) = O(n \log n \log \log n)$$

to describe the complexity for the multiplication of two n-bit integers.[8] Further, we set

$$\mathsf{M}(\mathbb{F}_p, n) = O(\mathsf{M}(n(\log p + \log n))), \quad \mathsf{M}(\mathbb{F}_q, n) = O(\mathsf{M}(\mathbb{F}_p, en)),$$
$$\mathsf{M}(R) = O(\mathsf{M}(\mathbb{F}_q, m))$$

where $q = p^e$, and $\mathsf{M}(R, d)$ denotes the complexity of the multiplication of two polynomials of degree d over the ring R. We also let $\mathsf{M}(R) = \mathsf{M}(R, 1)$ the complexity of the multiplication of two elements in R. These bounds can be obtained from Kronecker substitution [26]. Similarly, the multiplication of two polynomials of degree $O(d)$ over R costs

$$\mathsf{M}(R, d) = O(\mathsf{M}(\mathbb{F}_q, dm)) = O(\mathsf{M}(m\, e\, d(\log p + \log(m\, e\, d)))).$$

Moreover, we already note that later we will have $m \approx d \approx \ell$, $e \in \{1, 2\}$ and $\log(p) \in O(\log(\ell))$ for some integer ℓ, hence

$$\mathsf{M}(R) = O(\mathsf{M}(\ell \log \ell)) = O(\ell \log^2 \ell \log \log \ell),$$
$$\mathsf{M}(R, \ell) = O(\mathsf{M}(\ell^2 \log \ell)) = O(\ell^2 \log^2 \ell \log \log \ell).$$

Inversions in R can be computed with the same asymptotic complexity, for instance by using *Newton lifts*. Remarkably, this approach also allows us to lift roots of polynomials over k to roots of polynomials over R. Since Newton lifts play an important role in our algorithms, this standard method is presented in Algorithm 1.

[8]We remark that there exists an algorithm for integer multiplication in $O(n \log n)$ by Harvey and van der Hoeven, [20]. Here, we however use the bound $\mathsf{M}(n)$ defined above, in order to make it easier to compare our results to those in [8].

Algorithm 1 `Newton_lift`

```
Input: An element $\alpha \in k$ and a polynomial $f \in R[x]$ such that $f(\alpha) \equiv 0$ and $f'(\alpha) \not\equiv 0 \pmod{(\epsilon)}$.
Output: An element $\tilde{\alpha} \in R$ with $\tilde{\alpha} \equiv \alpha \pmod{(\epsilon)}$ and $f(\tilde{\alpha}) = 0$.
1: for $r \leftarrow 1, \dots, \lceil \log_2(m+1) \rceil$ do
2:     $\alpha \leftarrow \alpha + O(\epsilon^{\min(2^r, m+1)})$
3:     $\alpha \leftarrow \alpha - \frac{f(\alpha)}{f'(\alpha)}$
4: end for
5: return $\alpha$
```

3.2. **Lifting torsion points.** Let A be an abelian variety over k. Given a point $P \in A(k)$, and an m-th order deformation $\mathcal{A}$ of A, there exist multiple points $\tilde{P} \in \mathcal{A}(R)$ reducing to P. On the other hand, there is an isomorphism $\mathcal{A}[N] \cong A[N]$ if $p \nmid N$. In order to be able to lift isogenies, we need to make this isomorphism explicit. The case of elliptic curves is covered by Algorithm 2 and the special case of abelian surfaces and $N = 2$ is covered by Algorithm 3.

Lemma 3.1. *On input a point $P \in E[N]$ with N coprime to char(k), E an elliptic curve over k and an m-th order deformation $\mathcal{E}$, Algorithm 2 returns the unique lift $\tilde{P} \in \mathcal{E}[N]$ of P. For fixed N, the algorithm runs in time $O(1)$ over $R = k[\epsilon]/(\epsilon^{m+1})$.*

Proof. Since char$k \nmid N$, there is a unique lift $\tilde{P} = (\tilde{x}, \tilde{y})$ of $P = (x_0, y_0) \in E[N]$ which is also an N-torsion point. Let $f_{\mathcal{E},N}$ denote the N-th division polynomial of $\mathcal{E}$. Then $f_{\mathcal{E},N}(x_0) \equiv 0 \pmod{(\epsilon)}$, and we can lift x_0 to a root $\tilde{x}_0$ of $f_{\mathcal{E},N}$ using the method `Newton_lift` (Algorithm 1). The corresponding y-coordinate $\tilde{y}_0$ lying above $y_0 \in k$ is computed by another call to `Newton_lift`.

It is clear that $\tilde{P} = (\tilde{x}_0, \tilde{y}_0)$ is in $\mathcal{E}[N]$. And the running time of the algorithm is determined by the running time of `Newton_lift`. □

Algorithm 2 `lift_point`

```
Input: Elliptic curve $E$ over $k$, a point $P \in E(k)[N]$ with $p \nmid N$, and an $m$-th order deformation $\mathcal{E} : y^2 = f(x)$.
Output: The lift $\tilde{P} \in \mathcal{E}[N]$ of $P$.
1: $(x_0, y_0) \leftarrow P$
2: $f_{\mathcal{E},N}$ the $N$-th division polynomial of $\mathcal{E}$.
3: $\tilde{x}_0 \leftarrow$ Newton_lift$(x_0, f_{\mathcal{E},N})$
4: $\tilde{y}_0 \leftarrow$ Newton_lift$(y_0, y^2 - f(\tilde{x}_0))$
5: return $(\tilde{x}_0, \tilde{y}_0) \in \mathcal{E}[N]$
```

There are different ways to generalise Algorithm 2 to higher dimensions. In our applications, we only need to lift 2-torsion points of abelian surfaces. On decomposable abelian surfaces, we may just apply Algorithm 2 on every component. Algorithm 3 describes a general method for lifting 2-torsion points on abelian surfaces. For simplicity, we represent 2-torsion points as pairs $P = \{\alpha_1, \alpha_2\} \in A(k)[2]$. If $A = \text{Jac}(C)$, this means $P = [(\alpha_1, 0) + (\alpha_2, 0) - \infty_+ - \infty_-]$; and if $A = E_1 \times E_2$, it means $P = ((\alpha_1, 0), (\alpha_2, 0))$.

Algorithm 3 `lift_2_torsion`

Input: A principally polarised abelian surface A, a point $P \in A(k)[2]$, and an m-th order deformation $\mathcal{A}$.
Output: The lift $\tilde{P} \in \mathcal{A}[2]$ of P.

```
1: $\{\alpha_1, \alpha_2\} \leftarrow P$
2: if $A = \mathrm{Jac}(C)$ with $C : y^2 = f(x)$ then
3:     $\tilde{f} \leftarrow$ a lift of $f$ with $\mathcal{C} : y^2 = \tilde{f}(x)$ and $\mathcal{A} = \mathrm{Jac}(\mathcal{C})$
4:     $\tilde{f}_i \leftarrow \tilde{f}$ for $i = 1, 2$
5: else
6:     $A = E_1 \times E_2$ with $E_1 : y_1^2 = f_1(x_1)$ and $E_2 : y_2^2 = f_2(x_2)$
7:     if $\mathcal{A} = \mathcal{E}_1 \times \mathcal{E}_2$ then
8:         $\tilde{f}_i \leftarrow$ lift of $f_i$ with $\mathcal{E}_i : y_i^2 = \tilde{f}_i(x_i)$ for $i = 1, 2$
9:     else
10:        $\mathcal{A} = \mathrm{Jac}(\mathcal{C})$ with $\mathcal{C} : y^2 = \tilde{f}(x)$
11:        $\tilde{f}_i \leftarrow$ lift of $f_i$ with $\mathcal{C} : y_i^2 = \tilde{f}_i(x_i)$ for $i = 1, 2$
12:    end if
13: end if
14: $\tilde{\alpha}_i \leftarrow$ Newton_lift$(\alpha_i, \tilde{f}_i)$ for $i = 1, 2$
15: return $\{\tilde{\alpha}_1, \tilde{\alpha}_2\} \in \mathcal{A}[2]$
```

Lemma 3.2. *On input a point $P \in A[2]$, where A is a principally polarized abelian surface over k, and an m-th order deformation $\mathcal{A}$, Algorithm 2 returns the unique lift $\tilde{P} \in \mathcal{A}[2]$ of P. The algorithm runs in time $O(1)$ over $R = k[\epsilon]/(\epsilon^{m+1})$.*

Proof. First assume that A is indecomposable, i.e. $A = \mathrm{Jac}(C)$ with $C : y^2 = f(x)$. Then $\mathcal{A} = \mathrm{Jac}(\mathcal{C})$ and $\mathcal{C} : y^2 = \tilde{f}(x)$ is a deformation of C. In particular, lifting the torsion point P to $\mathrm{Jac}(\mathcal{C})$ consists in lifting the roots α_1, α_2 of f to roots of the polynomial $\tilde{f} \in R[x]$. Note that we need to choose an equation $y^2 = \tilde{f}(x)$ so that $\tilde{f} \equiv f \pmod{(\epsilon)}$.[9]

On the other hand, if $A = E \times E'$, there are two cases to consider. If $\mathcal{A} = \mathcal{E} \times \mathcal{E}'$ is also a product of elliptic curves, then the situation is similar as above. We simply lift roots α_1, α_2 to roots of the defining polynomials for $\mathcal{E}$ and $\mathcal{E}'$ respectively.

The case where $\mathcal{A} = \mathrm{Jac}(\mathcal{C})$ is indecomposable, but $A = E_1 \times E_2$ is a product of elliptic curves is more subtle. For $i \in \{1, 2\}$, we write $E_i : y_i^2 = f_i(x_i)$. Note that there necessarily exists a relation between the coordinates (x_1, y_1) and (x_2, y_2). In particular, there exist polynomials $\tilde{f}_i$ (of degree 5 or 6) in $k[\epsilon]/(\epsilon^{m+1})[x]$ such that the reduction modulo (ϵ) is equal to f_i. Consequently, the root α_i can be lifted to a root of $\tilde{f}_i$ using `Newton_lift`. Note that in order to interpret the resulting representation $\{\tilde{\alpha}_1, \tilde{\alpha}_2\} \in \mathcal{A}[2]$ of the 2-torsion point correctly, one needs to consider the polynomials $\tilde{f}_1$ and $\tilde{f}_2$. □

3.3. **Lifting an isogeny diamond.** Given an isogeny diamond over a field k as in Definition 2.11, there are two ways to lift it to R. The straight-forward method is to lift the individual elliptic curve isogenies to isogenies over R. Here, we describe a different method (Algorithm 4), where the product isogeny induced by Kani's

[9] In our applications (e.g. Algorithm 5), this is automatically the case.

Lemma is directly lifted to R. The key ingredient for this algorithm is Corollary 2.15.

Algorithm 4 `lift_isogeny_diamond`

Input: Elliptic curves E, E_a, E_b, E_{ab} which are the vertices of an isogeny diamond, and a group $K \subset E \times E_{ab}$ which is the kernel of the corresponding product isogeny. An m-th order deformation $\mathcal{E}$ of E.
Assertion: $j(E), j(E_a), j(E_b), j(E_{ab}) \notin \{0, 1728\}$.
Output: The lifted isogeny diamond $(\mathcal{E}, \mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_{ab})$ over R.

```
1: $\mathcal{E}_{ab} \leftarrow E_{ab}$, $\tilde{K} \leftarrow K$
   /*The precision is doubled at each step.*/
2: for $r \leftarrow 1, \dots, \lceil \log_2(m+1) \rceil$ do
3:     $j_{ab} = j(\mathcal{E}_{ab})$
4:     $R \leftarrow k[\epsilon]/\left(\epsilon^{\min(2^r, m+1)}\right)$
       /*We compute $\chi_{10}$ for two different lifts of $E_{ab}$.*/
5:     for $i = 0, 1$ do
6:         $j_i \leftarrow R(j_{ab}) + i \cdot \epsilon^{2^{r-1}}$
7:         $\mathcal{E}_i \leftarrow$ deformation($E_{ab}$) with $j(E_i) = j_i$ over $R$.
8:         $K_i \leftarrow$ lift($\tilde{K}$) $\subset (\mathcal{E} \times \mathcal{E}_i)[N]$.
9:         $A_i \leftarrow (\mathcal{E} \times \mathcal{E}_i)/K_i$.
10:        $\delta_i \leftarrow \chi_{10}(A_i)$
11:    end for
       /*The correct lift of $\mathcal{E}_{ab}$ is deduced from $\delta_0, \delta_1$.*/
12:    $j_{ab} \leftarrow j_{ab} - \frac{\delta_0}{\delta_1 - \delta_0}$
13:    $\mathcal{E}_{ab} \leftarrow$ deformation($E_{ab}$) with $j(\mathcal{E}_{ab}) = j_{ab}$
14:    $\tilde{K} \leftarrow$ lift($\tilde{K}$) $\subset (E \times E'')[N]$
15: end for
    /*We compute the lifted product isogeny.*/
16: $\mathcal{E}_a \times \mathcal{E}_b \leftarrow (\mathcal{E} \times \mathcal{E}_{ab})/\tilde{K}$.
17: return $(\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_{ab})$
```

Proposition 3.3. *Let E, E_a, E_b, E_{ab} be elliptic curves which are the vertices of an isogeny diamond, and let $K \subset E \times E_{ab}$ the kernel of the product isogeny induced by Kani's Lemma. Further assume that none of the elliptic curves have extra automorphisms, that is we assume $j(E), j(E_a), j(E_b), j(E_{ab}) \notin \{0, 1728\}$. On input this data together with an m-th order deformation $\mathcal{E}$ of E, Algorithm 4 outputs the deformations $(\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_{ab})$ such that F lifts to an isogeny $\tilde{F} : \mathcal{E} \times \mathcal{E}_{ab} \to \mathcal{E}_a \times \mathcal{E}_b$.*

Proof. The main part of the algorithm consists in computing the correct deformation of E_{ab}. Once this is done, we can lift the kernel K to a subgroup of $\mathcal{E} \times \mathcal{E}_{ab}$, and then $\mathcal{E}_a \times \mathcal{E}_b$ is computed as the codomain of the product isogeny with this kernel (Line 16).

To understand the main part, recall from Corollary 2.15 that there exists a polynomial $h \in k[x_1, x_2]$ with the property that

$$h(j(\mathcal{E}) - j(E), j(\mathcal{E}') - j(E_{ab})) = \chi_{10}(\mathcal{A}),$$

for any arbitrary deformation $\mathcal{E}'$ of E_{ab} and where $\mathcal{A}$ is the corresponding codomain of the lifted isogeny $F : \mathcal{E} \times \mathcal{E}' \to \mathcal{A}$. Our goal is to find the correct deformation

$\mathcal{E}' = \mathcal{E}_{ab}$ so that $\mathcal{A}$ is again decomposable, i.e. $\chi_{10} = 0$. If we knew the polynomial h, we could simply apply the Newton method to find the correct value for $j(\mathcal{E}_{ab})$. Since this is not the case, we use an interpolation based approach, where we evaluate h at two different values of $j(\mathcal{E}')$. Similar as in Newton's method, this approach lets us double the precision at each step.

To describe the idea in more detail, assume we are at Step r of the algorithm. Let $E_{ab,r-1}$ be the correct deformation of order $2^{r-1} - 1$, that is

$$h(j(\mathcal{E}) - j(E), j(E_{ab,r-1}) - j(E_{ab})) \equiv 0 \pmod{(\epsilon^{2^{r-1}})}.$$

Now, we compute two different deformations of E_{ab} of order $2^r - 1$, and denote them by E_0 and E_1. More precisely, we choose the lifts with j-invariants

$$j(E_0) = j(E_{ab,r-1}), \quad j(E_1) = j(E_{ab,r-1}) + \epsilon^{2^{r-1}}.$$

Note that we can compute arbitrary deformations, since $j(E_{ab}) \neq 0, 1728$ by assumption (cf. Example 2.7). For both cases $i = 0, 1$, we lift the kernel K to a subgroup $K_i \subset \mathcal{E} \times E_i$ using the method `lift_point` described in Algorithm 2. Then we compute the codomains $A_i = (\mathcal{E} \times E_i)/K_i$ of the resulting isogenies, and $\delta_i = \chi_{10}(A_i)$. With $j(E_{ab,r}) = j(E_{ab,r-1}) - \delta_0/(\delta_1 - \delta_0)$, we find that

$$h(j(\mathcal{E}) - j(E), j(E_{ab,r}) - j(E_{ab})) \equiv 0 \pmod{(\epsilon^{2^r})}.$$

After computing the data corresponding to this deformation, one can proceed to Step $r + 1$. □

3.4. **Lifting (smooth-degree) isogenies.** Given an isogeny $f : A \to A'$ and a deformation $\mathcal{A}$ of A, Proposition 2.8 shows that there exists a unique lift $\tilde{f} : \mathcal{A} \to \mathcal{A}'$. Let $G \subset A[N]$ be the kernel of f. The straight-forward approach for lifting f consists in lifting the kernel generators, for instance using Algorithms 2, in order to obtain a lifted kernel $\tilde{G} \subset \mathcal{A}[N]$. Then the isogeny can be computed using standard algorithms. This is the approach used in Algorithm 4.

However, if N is composite the above approach is not optimal. For instance, in the case $N = 2^n$, an isogeny can be lifted in time $O(n)$ (provided some auxiliary data from the original isogeny), while the naive method has complexity $O(n \log(n))$. To keep everything explicit, we restrict to the case of $(2^n, 2^n)$-isogenies.

Algorithm 5 `lift_2_2_chain`

Input: A $(2^n, 2^n)$-isogeny $f = f_n \circ \cdots \circ f_1 : A_0 \to A_n$ over k, the data $\ker(f_i) = \langle P_{1,i}, P_{2,i} \rangle$ for each i, and an m-th order deformation $\mathcal{A}_0$ of A_0.
Output: The deformation $\mathcal{A}_n$ so that f lifts to an isogeny $\tilde{f} : \mathcal{A}_0 \to \mathcal{A}_n$.

```
1: for $i = 1, \ldots, n$ do
2:     $\tilde{P}_{j,i} \leftarrow$ lift_2_torsion $(A_{i-1}, P_{j,i}, \mathcal{A}_{i-1})$ for $j = 1, 2$
3:     $\mathcal{A}_i \leftarrow$ 2_2_isogeny$(\mathcal{A}_{i-1}, \langle \tilde{P}_{1,i}, \tilde{P}_{2,i} \rangle)$
4: end for
5: return $\mathcal{A}_n$
```

Lemma 3.4. *Given a $(2^n, 2^n)$-isogeny $f = f_n \circ \cdots \circ f_1 : \mathcal{A}_0 \to \mathcal{A}_n$ over a field k together with the data $\ker(f_i) = \langle P_{1,i}, P_{2,i} \rangle$ for each $i \in \{1, \ldots, n\}$, and an m-th order deformation $\mathcal{A}_0$ of A_0, Algorithm 5 outputs a deformation $\mathcal{A}_n$ of A_n so* *that f lifts to an isogeny $\tilde{f} : \mathcal{A}_0 \to \mathcal{A}_n$. The algorithm runs in time $O(n)$ over $R = k[\epsilon]/(\epsilon^{m+1})$.*

Proof. The isogeny chain computed in Algorithm 5 is correct, since at each step the kernel generators are lifted correctly as per Lemma 3.2. The method `2_2_isogeny` is used as a black box. It is only important that it runs in time $O(1)$ over R. □

Remark 3.5. An even faster method to lift a smooth degree isogeny is to use modular correspondences. For instance, a 2^n-isogeny over k is given by a sequence of modular points which are solutions of the modular correspondance $\overline{\Phi_2}$, and lifting the isogeny corresponds to lifting this sequence of points. In practice, since we want to be able to evaluate modular forms, we need to work with normalised isogenies, hence use a version of $\overline{\Phi_2}$ which keeps track of our differentials.

For instance, Richelot isogenies are well suited to write explicit modular equations. Explicit formulas are given by "multiradical isogeny" formulas [10, § 4.2] which describe the modular correspondence $\overline{\Phi_2} : \mathsf{A}_2(\Gamma_1(4)) \to \mathsf{A}_2(\Gamma_1'(4)) \to \mathsf{A}_2(\Gamma_1(2)) \times \mathsf{A}_2(\Gamma_1(2))$ (see [10, § 3.1] for the notations). The theta duplication formula is also naturally 2-radical in dimension 2. Using one of these radical 2-isogeny formulas, lifting each such 2-isogeny in dimension 2 amounts to lifting (via Newton) three square roots. In our implementation, we follow this approach. It has the advantage that it bypasses the need to lift torsion points and evaluate isogeny formulas on these lifted points. Another potential (constant-time) speed-up could be obtained by using 4-radical isogeny formulas instead.

3.5. **Deforming a general isogeny.** In this section we prove the general statement from Theorem 1.3. We are given an efficient representation of an isogeny $f : E_1 \to E_2$, and we want to lift it to $\tilde{f} : \mathcal{E}_1 \to \mathcal{E}_2$ for some given m-th order deformation $\mathcal{E}_1/R$.

Initialisation: We fix $N > \ell$ with N prime to ℓ and p, and such that the N-torsion is accessible (which means that its Sylow subgroups lie in small extensions of the base field), for instance N is powersmooth or $N = 2^n$ if the 2^n-torsion of E is rational. We write $N - \ell = a^2 + b^2 + c^2 + d^2$, and consider α, the 4×4 quaternion matrix with norm $N - \ell$. Then α induces an endomorphism on E_1^4 and E_2^4. We can construct the isogeny diamond

$$\begin{tikzcd} E_1^4 \arrow[r, "f"] \arrow[d, "\alpha"] & E_2^4 \arrow[d, "\alpha"] \\ E_1^4 \arrow[r, "f"] & E_2^4. \end{tikzcd}$$

which allows us to embed f into an 8-dimensional N-endomorphism $F : E_1^4 \times E_2^4 \to E_1^4 \times E_2^4$. By assumption, we have an efficient representation of f, hence we can evaluate it on the N-torsion in order to compute the HD representation F of f. See [39] for more details.

Since N is taken to be smooth, we can decompose F as a product of small degree isogenies. In theory, F can be computed in the theta model, using the isogeny algorithm of [30], working with level 4 theta functions, and representing our intermediate abelian varieties by their theta constants. The codomain of F is equal to $E_1^4 \times E_2^4$ but it may have different theta constants than the domain (because the level $\Gamma(4, 8)$-theta structure needs not be preserved), so we apply a symplectic change of basis at the end to get matching theta constants. A way to compute this matrix is described in [13, Appendix F].

Newton iterations: We then proceed by a Newton iteration, doubling the precision m at each stage. For simplicity we explain how to go from $m = 1$ to $m = 2$ here, the general case being similar.

Let $\tilde{f} : \mathcal{E}_1 \to \mathcal{E}_2$ be the deformation of f to $\mathcal{E}_1$. The isogeny diamond above certainly deforms (with the same matrix α), and so F deforms to an endomorphism $\tilde{F}$ of $\mathcal{E}_1^4 \times \mathcal{E}_2^4$. On the other hand, if we take an arbitrary lift $\mathcal{E}_2'$ of E_2 to R, the corresponding deformation $\tilde{F}'$ is only an isogeny. In fact, if $\tilde{F}'$ is an endomorphism, then it is given by a matrix, which reduces to the matrix giving F (by unicity of deformations), hence contains the deformation $\tilde{f}$.

In summary, $\mathcal{E}_2'$ is the correct codomain of the deformation of f if and only if the codomain of $\tilde{F}$ is equal to the domain. Using modular invariants J (e.g. the theta constants), this can be tested via the equality $J(\mathcal{E}_1^4 \times \mathcal{E}_2'^4) = J(\text{codomain}(\tilde{F}))$. By the same arguments as in Section 2, in particular Corollary 2.15, the left and right member of this equality are analytic in terms of the deformation parameter λ of $\mathcal{E}_2$ (i.e. are given by power series). Since we are doing a Newton iteration, everything becomes linear, so we just need to interpolate between two different evaluations of the deformation parameter λ to solve the equation. For each of these two choices of λ, we need to compute the deformation of $\tilde{F}$ to recover its codomain.

Computing $\tilde{F}$: We proceed as in Section 3.4. Since F is decomposed into a product of small isogenies over $\mathbb{F}_q$, we deform these isogenies step by step. The easiest way is to deform the points in the kernel and apply the theta isogeny algorithm.

Deforming a point P of N-torsion: We also proceed by a Newton iteration. Division polynomials are not easy to compute in higher dimension. Instead, we just rely on the fact that there is a unique deformation $\tilde{P}$ of P which is still of n-torsion, and that taking an arbitrary deformation $\tilde{P}'$ we can efficiently compute $n \cdot \tilde{P}'$. In other words, it is easy to evaluate the n-division polynomials, which is enough for the Newton iteration.

Complexity: Since we double the precision m at each step, and the arithmetic of R is super-linear in terms of m, the last Newton step is dominating.

The dominant step is in the computation of the deformation $\tilde{F}$; it consists in deforming $O(\log \ell)$ isogenies of degree $O(\log \ell)$ in the worst case (e.g. when N is powersmooth). The algorithm cost is thus polynomial in $\log \ell$ arithmetic operations (for deforming the generators of the kernels and then computing the isogeny) in R at precision m.

In the best case, we can take $N = 2^n$ and the N-torsion is rational in E_1, so we just need to deform $O(\log \ell)$ 2-isogenies.

4. Computing modular polynomials over finite fields

In this section, we present algorithms for computing the modular polynomial $\varphi_\ell(X, Y)$ over $\mathbb{F}_p$.[10] We describe very explicit versions of the algorithm for a family of primes p depending on ℓ. These methods will be used in our implementation for computing the modular polynomial over $\mathbb{Q}$. In addition, Section 4.4 discusses the generalization to arbitrary primes.

[10]We use the notation φ_ℓ for the modular polynomial over a finite field, and Φ_ℓ for the modular polynomial with coefficients in $\mathbb{Z}$.

In all cases, the main ingredient for our procedure is an algorithm to lift isogeny diamonds over $\mathbb{F}_p$ to isogeny diamonds over $\mathbb{F}_p[[\epsilon]]$ with precision at least $\ell+2$. The case distinction is necessary, since we construct (ℓ, d)-isogeny diamonds with $\ell + d$ smooth, which depends on the value of ℓ.

4.1. **Suitable primes.**

Depending on $\ell \pmod 4$, we define a set of primes $\mathcal{P}_\ell$ for which our algorithm can compute the modular polynomial over $\mathbb{F}_p$. For an odd prime ℓ, we set

$$\mathcal{P}_\ell = \{p > 11 \text{ prime} : \exists\ n, a, b \text{ with } 2^n - c_\ell \cdot \ell = a^2 + 4b^2 \text{ and } 2^n \cdot c_\ell \cdot \ell \mid p+1\},$$

where

$$c_\ell = \begin{cases} 1 & \text{if } \ell \equiv 3 \pmod 4, \\ 3 & \text{if } \ell \equiv 1 \pmod 4). \end{cases}$$

Heuristic 4.1. Let ℓ be a positive integer and c_ℓ as defined above. Then we expect

$$\#\{n \in \mathbb{N} \mid 2^n - c_\ell \cdot \ell = a^2 + 4b^2,\ n \leq x\} \approx \frac{x}{\sqrt{x}}.$$

Under Heuristic 4.1, we expect to find a value $n \in \mathcal{O}(\log(\ell))$. The intuition behind this heuristic and our choice for c_ℓ is explained in the remark below.

Remark 4.2. Recall that an element $z \in \mathbb{N}$ can be written as a sum of two squares, $z = a^2 + b^2$, if and only if its prime decomposition $z = \prod_i p_i^{k_i}$ contains no prime factor $p_i = 3 \pmod 4$ with k_i odd. In particular, an element $z \equiv 3 \pmod 4$ cannot be written as a sum of two squares. Now $c_\ell \in \{1, 3\}$ is chosen so that this necessary congruence condition is satisfied for all $z = 2^n - c_\ell \cdot \ell$.

Note that in our setting, we need to write $z = a^2 + 4b^2$. However, we are always in the case that z is odd, hence this is equivalent to z being a sum of two squares. The fraction of elements smaller than some x that can be written as a sum of two squares is known to be in $O\left(1/\sqrt{\log(x)}\right)$, hence

$$\#\{z = a^2 + 4b^2 \mid z \leq x\} \approx \frac{x}{\sqrt{\log(x)}}.$$

Essentially, Heuristic 4.1 means that we expect the elements of the form $a^2 + 4b^2$ to be uniformly distributed among the numbers of the form $2^n - c_\ell \cdot \ell$ for varying $n \in \mathbb{N}$.

Under Heuristic 4.1, we obtain that there are $\mathcal{O}(\ell)$ primes $p \in P_\ell$ that have about the same logarithmic size as ℓ, i.e. $\log(p)$ in $\mathcal{O}(\log(\ell))$. A more precise estimate is provided in the lemma below.

Lemma 4.3. *Let ℓ be a positive integer and $\mathcal{P}_\ell$ as defined above. Then*

$$\#\{p \in P_\ell \mid p \leq x\} \gtrapprox \frac{x}{2^n \ell \log(x)}.$$

More precisely, the m-th prime in P_ℓ is bounded by $(2^n\ell)^L m \log(m)^2$, where $L \leq 5$ is Linnik's constant. If Heuristic 4.1 holds, the bound is $(\ell)^{L'} m \log(m)^2$, for some L'.

Proof. We look at the smallest integer so that there exist integers a, b with $2^n - c_\ell \cdot \ell = a^2 + 4b^2$. For this triple (n, a, b), we can sieve through the primes to find those which satisfy

$$p \equiv -1 \pmod{2^n \cdot c_\ell \cdot \ell}.$$

It follows from the *prime number theorem for arithmetic progressions* that the proportion of primes of this form is $1/\varphi(2^n \cdot c_\ell \cdot \ell) \approx 1/\ell^2$ (with φ denoting Euler's totient function).

For the more precise statement, we need an upper bound on the smallest suitable prime. We need to find primes congruent to -1 modulo $A = 2^n \cdot \ell$. By Linnik's theorem [21, Corollary 18.8], the m-th such prime is bounded by $A^L m \log(m)^2$ where $L \leq 5$ is Linnik's constant. Under Heuristic 4.1, we have $2^n \leq \ell^{\mathfrak{D}}$ for some constant $\mathfrak{D}$, which concludes the proof. □

4.2. **The case $\ell \equiv 3 \pmod 4$.** Here, we present an algorithm for computing the modular polynomial φ_ℓ over $\mathbb{F}_p$ when $\ell \equiv 3 \pmod 4$ for primes p in the set $\mathcal{P}_\ell$.

Algorithm 6 `modular_polynomial_modp` $\ell \equiv 3 \pmod 4$

Input: A prime $\ell \equiv 3 \pmod 4$, and a prime $p \in \mathcal{P}_\ell$.
Output: The modular polynomial $\varphi_\ell(X,Y) \in \mathbb{F}_p[X,Y]$.

```
    /*Setup*/
 1: Set $E : y^2 = x^3 + 6x^2 + x$ over $\mathbb{F}_{p^2}$
 2: $\iota \leftarrow \iota \in End(E)$ with $\iota \circ \iota = [-4]$
 3: $\gamma \leftarrow [a] + [b]\iota$, where $2^n - \ell = a^2 + 4b^2$ for some $n$.
 4: Compute $P_{2^n}, Q_{2^n}$ with $E[2^n] = \langle P_{2^n}, Q_{2^n} \rangle$
 5: Compute $P_\ell, Q_\ell$ with $E[\ell] = \langle P_\ell, Q_\ell \rangle$
 6: $\tilde{j} \leftarrow j(E) + \epsilon \in \mathbb{F}_{p^2}[\epsilon]/(\epsilon^{\ell+2})$
 7: $\mathcal{E} \leftarrow$ elliptic curve with $j(\mathcal{E}) = \tilde{j}$
    /*computing and lifting all $\ell$-isogenies*/
 8: for $k \leftarrow 0, \ldots, \ell$ do
 9:     if $k = \ell$ then
10:         $P_k \leftarrow Q_\ell$
11:     else
12:         $P_k \leftarrow P_\ell + k \cdot Q_\ell$
13:     end if
14:     $E_k \leftarrow E/\langle P_k \rangle$ with $f_k : E \to E_k$
        /*constructing the $(\ell, 2^n - \ell)$-isogeny diamond*/
15:     $E'_k \leftarrow E/\langle \gamma(P_k) \rangle$ with $f'_k : E \to E'_k$
16:     $K \leftarrow \langle (\hat{\gamma}(P_{2^n}), f'_k(P_{2^n})), (\hat{\gamma}(Q_{2^n}), f'_k(Q_{2^n})) \rangle$
17:     $(\mathcal{E}, \mathcal{E}_k, \mathcal{E}', \mathcal{E}'_k) \leftarrow$ lift_isogeny_diamond$(E, E_k, E, E'_k, K, \mathcal{E})$
18:     $\tilde{j}_k \leftarrow j(\mathcal{E}_k)$
19: end for
    /*final step*/
20: $\varphi \leftarrow \prod_{k=0}^{\ell} (Y - \tilde{j}_k)(\epsilon = X - j(E)) \in \mathbb{F}_p[X,Y]$
21: return $\varphi$
```

Theorem 4.4. *On input a prime $\ell \equiv 3 \pmod 4$ and a prime $p \in \mathcal{P}_\ell$, Algorithm 6 returns the modular polynomial φ_ℓ over $\mathbb{F}_p$. The algorithm runs in*

$$O(\log p + \ell^2) \cdot \mathsf{M}(\mathbb{F}_{p^2}) + O(n\ell) \cdot \mathsf{M}(R) + O(\log \ell)\mathsf{M}(R, \ell),$$

with $R = \mathbb{F}_{p^2}[\epsilon]/(\epsilon^{\ell+2})$, and $\mathsf{M}(\cdot)$ as defined in Subsection 3.1.

Proof. First, the algorithm sets $E : y^2 = x^3 + 6x^2 + x$ over $\mathbb{F}_{p^2}$. This curve is 2-isogenous to the elliptic curve with j-invariant 0, hence there exists an endomorphism $\iota : E \to E$ with the property $\iota^2 = [-4]$. Since $p \in P_\ell$, there exist values n, a, b so that $2^n - \ell = a^2 + 4b^2$, and the algorithm sets $\gamma = [a] + [b]\iota$ which is an endomorphism of degree $2^n - \ell$. Further, a basis (P_{2^n}, Q_{2^n}) for $E[2^n]$ and a basis (P_ℓ, Q_ℓ) for $E[\ell]$ are computed. Note that by assumption $p \equiv 3 \pmod 4$, hence the elliptic curve E is supersingular and has cardinality $(p+1)^2$. As a consequence, the $\ell \cdot 2^n$-torsion is $\mathbb{F}_{p^2}$-rational and it is not necessary to work in field extensions. The cost of the setup is dominated by the cost for computing the N-torsion bases for $N \in \{\ell, 2^n\}$. This can be done by sampling two points $P, Q \in E(\mathbb{F}_{p^2})$, computing $P_N = \frac{p+1}{N} \cdot P$, $Q_N = \frac{p+1}{N} \cdot Q$ and the Weil pairing $e_N(P_N, Q_N)$. This is repeated until $e_N(P_N, Q_N)$ has order N.[11] This step costs

$$(\log(p) + \log \ell) \cdot \mathsf{M}(\mathbb{F}_{p^2}) \tag{4}$$

In the last line of the setup, we compute the deformation $\mathcal{E}$ of E with j-invariant $j(\mathcal{E}) = j(E) + \epsilon$. Note that $j(E) = (2 \cdot 3 \cdot 11)^3 \notin \{0, 1728\}$ for $p > 11$, hence such a deformation exists, and it can be computed in time $O(1)$ in $R = \mathbb{F}_{p^2}[\epsilon]/(\epsilon^{\ell+2})$. This means the complexity of this step is simply

$$O(1) \cdot \mathsf{M}(R). \tag{5}$$

The main part of the algorithm consists of computing the $\ell + 1$ different ℓ-isogenies emanating from the elliptic curve E; and lifting these to isogenies emanating from the elliptic curve $\mathcal{E}$ with j-invariant $\tilde{j} = j + \epsilon$. Using the square-root Vélu Algorithm, the isogeny computations over the ground field can be done in $\tilde{O}(\ell\sqrt{\ell})$ $\mathbb{F}_{p^2}$-multiplications. For the overall analysis it is sufficient to use the standard Vélu formulas, which results in

$$O(\ell^2) \cdot \mathsf{M}(\mathbb{F}_{p^2}). \tag{6}$$

To explain the lifting step in more detail, assume that we are at step k and want to lift the ℓ-isogeny $f_k : E \to E_k$ to an isogeny with domain $\mathcal{E}$. To this end, we construct an $(\ell, 2^n - \ell)$-isogeny diamond as depicted below.

$$\begin{tikzcd} E \arrow[r, "f_k"] \arrow[d, "\gamma"] & E_k \arrow[d, "\gamma'"] \\ E \arrow[r, "f'_k"] & E'_k \end{tikzcd}$$

This gives rise to a product isogeny $F : E \times E'_k \to E \times E_k$ with kernel

$$K = \langle (\hat{\gamma}(P_{2^n}), f'_k(P_{2^n})), (\hat{\gamma}(Q_{2^n}), f'_k(Q_{2^n})) \rangle.$$

On input the vertices of the isogeny diamond E, E_k, E, E'_k and the lift $\mathcal{E}$ together with the kernel K, the method `lift_isogeny_diamond` (Algorithm 4) outputs the vertices of the lifted isogeny diamond $(\mathcal{E}, \mathcal{E}_k, \mathcal{E}', \mathcal{E}'_k)$. In particular, this contains the lift $\mathcal{E}_k$ of E_k such that f_k lifts to an isogeny $\tilde{f}_k : \mathcal{E} \to \mathcal{E}'_k$ over $\mathbb{F}_{p^2}[\epsilon]/(\epsilon^{\ell+2})$. We save its j-invariant $\tilde{j}_k$. Note that since we are lifting a $(2^n, 2^n)$-chain here, we can use an improved lifting strategy in `lift_isogeny_diamond` as described in Subsection

[11] There are better methods to compute a basis in practice. In particular, in the case of $N = 2^n$, one should use the methods developed in the framework of SIDH key compression [12].

3.4. For each ℓ-isogeny this gives us complexity $O(n)$ in R, hence overall the lifting step for all ℓ-isogenies costs

$$O(n\ell) \cdot \mathsf{M}(R) \tag{7}$$

In the final step, the product

$$\varphi_\ell(j(E) + \epsilon, Y) = \prod_{k=0}^{\ell} (Y - \tilde{j_k}) \in R[Y].$$

is computed and evaluated at $\epsilon = X - j(E)$. This yields the modular polynomial $\varphi_\ell(X, Y)$ (modulo $(X - j(E))^{\ell+2}$). Given that $\varphi_\ell(X, Y)$ has degree $\ell + 1$, we may ignore the modulus and we conclude that the output is the modular polynomial $\varphi_\ell(X, Y)$ in $\mathbb{F}_p$. The evaluation of the product can be done via a product-tree strategy, which costs $O(\log \ell)\mathsf{M}(R, \ell)$.In our setting, the complexity for this multiplication is simply given by

$$O(\log \ell)\mathsf{M}(R, \ell) \tag{8}$$

Taking the sum over the complexities from Equations 4 - 8, we obtain a complexity of

$$O(\log p + \ell^2) \cdot \mathsf{M}(\mathbb{F}_{p^2}) + O(n\,\ell) \cdot \mathsf{M}(R) + O(\log \ell)\mathsf{M}(R, \ell).$$

□

4.3. **Generalization to $\ell \equiv 1 \pmod 4$.** In Algorithm 6 we described a method to compute the modular polynomial φ_ℓ in $\mathbb{F}_p[X, Y]$ for primes in P_ℓ, when $\ell \equiv 3 \pmod 4$. In this part, we sketch a modification of the algorithm which allows us to compute the modular polynomial for $\ell \equiv 1 \pmod 4$.

The different treatment stems from the fact that in this case, there do not exist integers a, b, n so that $2^n - \ell = a^2 + 4b^2$. Instead we consider the equation $2^n - 3 \cdot \ell = a^2 + 4b^2$, and extend the ℓ-isogenies by an auxiliary 3-isogeny.

Theorem 4.5. *There exists a modification of Algorithm 6 which on input a prime $\ell \equiv 1 \pmod 4$ and a prime $p \in \mathcal{P}_\ell$, returns the modular polynomial φ_ℓ over $\mathbb{F}_p$. The algorithm runs in*

$$O(\log p + \ell^2) \cdot \mathsf{M}(\mathbb{F}_{p^2}) + O(n\ell) \cdot \mathsf{M}(R) + O(\log \ell) \cdot \mathsf{M}(R, \ell),$$

with $R = \mathbb{F}_{p^2}(\epsilon)/(\epsilon^{\ell+2})$, and $\mathsf{M}(\cdot)$ as defined in Subsection 3.1.

Proof. We outline the modifications of Algorithm 6. It will be clear that these changes have no effect on the asymptotic complexity of the algorithm.

In the setup, we choose an endomorphism γ as $\gamma = [a] + [b]\iota$ with $2^n - 3 \cdot \ell = a^2 + 4b^2$. And we also compute an auxiliary degree-3 isogeny $g : E \to C_0$.

The main change occurs in the construction of the isogeny diamond. In order to make γ and the ℓ-isogeny f_k fit into an isogeny diamond (with smooth induced product isogeny), it is necessary to expand f_k by the auxiliary isogeny g to obtain the following configuration.

$$\begin{tikzcd} C_0 \arrow[d, "\gamma_0"] & E \arrow[l, "g"'] \arrow[r, "f_k"] \arrow[d, "\gamma"] & E_k \arrow[d, "\gamma'"] \\ C_1 & E \arrow[l, "g'"] \arrow[r, "f_k'"] & E_k' \end{tikzcd} \tag{9}$$

The outer square is a $(3 \cdot \ell, 2^n - 3 \cdot \ell)$-isogeny diamond and induces the product isogeny $F : C_0 \times E'_k \to C_1 \times E_k$ with kernel

$$K = \left\langle \left(\hat{\gamma_0}(P_{2^n}), f'_k \circ \hat{g}'(P_{2^n}) \right), \left(\hat{\gamma_0}(Q_{2^n}), f'_k \circ \hat{g}'(Q_{2^n}) \right) \right\rangle,$$

where (P_{2^n}, Q_{2^n}) is a basis for $C_1[2^n]$.

Recall that the goal is to lift the ℓ-isogeny f_k, for some fixed lift $\mathcal{E}$ of E. For this purpose, one first computes the lift $\tilde{g} : \mathcal{E} \to \mathcal{C}_0$ explicitly. Then we call `lift_isogeny_diamond` on input $(C_0, E_k, C_1, E'_k, K, \mathcal{C}_0)$. The output contains the lift $\mathcal{E}_k$, and the algorithm proceeds as in the case $\ell \equiv 3 \pmod 4$. □

The modified version of Algorithm 6 which is outlined in the proof above, is presented in Appendix A (Algorithm 8).

4.4. **The general case.** In this section, we give a quick overview of the proof of Corollary 1.4. If we start with an HD representation of the ℓ-isogenies starting from E_0, we can deform them to $k[\epsilon]/(\epsilon^{m+1})$ using Theorem 1.3 as explained in Section 3.5. When $m > \ell$, this is enough to reconstruct $\Phi_\ell \mod p$ in quasi-linear time as in the proof of Theorem 4.4.

In the previous two subsections, we provided algorithms to compute the modular polynomial φ_ℓ over a prime field $\mathbb{F}_p$, where p was an element in $\mathcal{P}_\ell$. The number of primes of this form relies on Heuristic 4.1 which describes the size of n, for which $2^n - \ell$ can be written as a sum of two squares. Using a dimension 8 embedding instead of a dimension 2 embedding, as explained in Section 3.5, we generalize the techniques to a larger set of primes by expressing $2^n - \ell$ as a sum of four squares. More explicitly, this results in an algorithm for all primes p in

$$\mathcal{P}^*_\ell = \{p > 11 \text{ prime} : 2^n \cdot \ell \mid p + 1, \text{ where } n = \lceil \log_2(\ell) \rceil\}$$

of complexity $O(\log^2 p + \ell^2 \log p)$.

5. Computing the modular polynomial

Here, we present our algorithm for computing the modular polynomial $\Phi_\ell \in \mathbb{Z}[X, Y]$ of elliptic curves. The procedure is summarized in Algorithm 7. It is a CRT based approach, where the modular polynomial is computed modulo many small primes using Algorithm 6 or its modification described in Subsection 4.3. The runtime of the algorithm relies on Heuristic 4.1, but we prove that there also exists an unconditional version with the same asymptotic runtime.

Theorem 5.1. *On input an odd prime ℓ, Algorithm 7 computes the modular polynomial $\Phi_\ell \in \mathbb{Z}[X, Y]$. Under Heuristic 4.1, the algorithm runs in time*

$$O(\ell^3 \log^3 \ell \log \log \ell).$$

Proof. The bound

$$B = 6\ell \log(\ell) + 16\ell + \min(2\ell, 14\sqrt{\ell} \log \ell) + \log(2)$$

in the first line of the algorithm is equal to the bound on the logarithmic height of the coefficients of Φ_ℓ (see Eq. 1) plus $\log(2)$. Consequently, the coefficients of Φ_ℓ are uniquely determined by their residues modulo $\exp(B)$.

In Line 4, a minimal integer n is chosen for which there exists a pair (a, b) so that $2^n - c_\ell \cdot \ell = a^2 + 4b^2$. Under Heuristic 4.1, we may assume that $n = O(\log(\ell))$.

Algorithm 7 `modular_polynomial`

```
Input: An odd prime $\ell$.
Output: The modular polynomial $\Phi_\ell(X,Y) \in \mathbb{Z}[X,Y]$.
 1: $B \leftarrow 6\ell\log(\ell) + 16\ell + \min(2\ell, 14\sqrt{\ell}\log\ell) + \log(2)$
 2: $P \leftarrow 1$, $\Phi_\ell \leftarrow 0$.
 3: $c_\ell \leftarrow -\ell \pmod 4$.
 4: $n \leftarrow \min\{n \mid \exists\, a, b : 2^n - c_\ell \cdot \ell = a^2 + 4b^2\}$
 5: while $P < \exp(B)$ do
 6:     $p \leftarrow$ next prime with $2^n \cdot c_\ell \cdot \ell \mid p+1$
 7:     $\varphi_\ell \leftarrow$ modular_polynomial_modp$(\ell, p) \in \mathbb{F}_p[X,Y]$
 8:     $P \leftarrow P \cdot p$
 9:     $\Phi_\ell \leftarrow$ CRT$(\Phi_\ell, \varphi) \in \mathbb{Z}/P\mathbb{Z}[X,Y]$
10: end while
11: return $\Phi$
```

In the main part of the algorithm, the modular polynomial $\varphi_\ell \in \mathbb{F}_p[X,Y]$ is computed for various suitable primes $p \in P_\ell$, until the modulos P is at least $\exp(B)$. At each step, we update $P = P \cdot p$ and compute $\Phi_\ell \in \mathbb{Z}/P\mathbb{Z}$ using an explicit version of the Chinese Remainder Theorem.

As per Theorems 4.4 and 4.5, the computation of $\varphi_\ell \in \mathbb{F}_p[X,Y]$ is done in time

$$O(\log p + \ell^2) \cdot \mathsf{M}(\mathbb{F}_{p^2}) + O(n\ell) \cdot \mathsf{M}(R) + O(\log \ell)\mathsf{M}(R, \ell),$$

with $R = \mathbb{F}_{p^2}(\epsilon)/(\epsilon^{\ell+2})$. Further, we may assume that $n = O(\log(\ell))$ under Heuristic 4.1. Note that $B = O(\ell\log(\ell))$, hence it suffices to compute φ_ℓ for $O(p)$ many primes of size $\log(p) \in O(\log(\ell))$. Lemma 4.3 assures that there are enough primes of the desired form in $\mathcal{P}_\ell$. In this setting,

$$\mathsf{M}(\mathbb{F}_{p^2}) = \mathsf{M}(\log\ell) = O(\log\ell\log\log\ell\log\log\log\ell),$$
$$\mathsf{M}(R) = \mathsf{M}(\ell\log\ell) = O(\ell\log^2\ell\log\log\ell),$$
$$\mathsf{M}(R,\ell) = \mathsf{M}(\ell^2\log\ell) = O(\ell^2\log^2\ell\log\log\ell)$$

In conclusion, we obtain

$$O(\ell^3\log^3\ell\log\log\ell)$$

for the overall runtime of Algorithm 7. □

Theorem 5.2. *There exists an algorithm for computing the modular polynomial Φ_ℓ for any prime ℓ, with unconditional runtime*

$$O(\ell^3\log^3\ell\log\log\ell).$$

Proof. We use the same proof as in Theorem 5.1, with the exception that P_ℓ is replaced by the set of primes $\mathcal{P}_\ell^*$ defined in Section 4.4. This requires to use dimension 8 embeddings instead of dimension 2 embeddings. However, the advantage of $\mathcal{P}_\ell^*$ is that we can set $n = \lceil\log_2(\ell)\rceil$ in the statement of Lemma 4.3 without heuristic assumptions. □

Remark 5.3. The runtime of the algorithms from Theorem 5.1 and 5.2 improves to $O(\ell^3\log^3\ell)$, when the bound $\mathsf{M}(n) = O(n\log n)$ from [20] is applied to describe the multiplication of n-bit integers, see also Subsection 3.1.

6. Implementation

A proof-of-concept implementation of Algorithm 7 (with $\ell \equiv 3 \pmod 4$) in SageMath [45] is available in our GitHub repository [28]. The repository further contains the different subroutines for computing with deformations of elliptic curves and isogenies presented in this paper.

Our implementation works with elliptic curves in Montgomery form, and we use the available functions in SageMath to compute elliptic curve isogenies. The computation of $(2^n, 2^n)$-isogenies is based on the formulas from [27], enhanced by explicit formulas for splitting and gluing isogenies. The individual $(2, 2)$-isogenies in this framework are naturally represented by radical formulas which facilitates an efficient computation of the deformation of the isogeny chains.

In the future, we plan to switch to the faster theta formulas from [14] to compute the $(2^n, 2^n)$-isogenies and then use radical $(2, 2)$-isogenies in the theta model for computing the deformation of these isogenies. However, to achieve a fast running time, we also require a dedicated implementation with a fast arithmetic for R and fast polynomial multiplication over R, and that remains a future work.

A last remark is that we only require to compute ℓ-isogenies in dimension 1 for the initialisation step: the lifting step is done through 2^n-isogenies in dimension 2. We have seen that the initialisation step is not dominant, even if we use the Vélu formula rather than the sqrt-Vélu algorithm. Still, we remark that one could also do the initialisation step using only 2^n-isogenies in dimension 2, using the Clapotis framework [35] to convert reduced ideals of norm ℓ to isogenies, as is done in [2]. As a fun side effect, this would relax the condition that $\ell \mid p^2 \pm 1$, and would allow to compute ϕ_ℓ while bypassing entirely Vélu's formulas for dimension 1 ℓ-isogenies.

References

[1] Arthur OL Atkin. The number of points on an elliptic curve modulo a prime. *preprint*, 1988.

[2] Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West: The fast, the small, and the safer. Cryptology ePrint Archive, Paper 2024/760, 2024.

[3] Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: fast encryption from supersingular torsion attacks. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 98–126. Springer, 2023.

[4] Daniel Bernstein and Jonathan Sorenson. Modular exponentiation via the explicit chinese remainder theorem. *Mathematics of Computation*, 76(257):443–454, 2007.

[5] Daniel J Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *Open Book Series*, 4(1):39–55, 2020.

[6] Florian Breuer, Desirée Gijón Gómez, and Fabien Pazuki. Explicit bounds on the coefficients of the modular polynomials and the size of $X\_0(n)$. arXiv preprint arXiv:2310.14428, 2023.

[7] Reinier Bröker. Constructing supersingular elliptic curves. *J. Comb. Number Theory*, 1(3):269–273, 2009.

[8] Reinier Bröker, Kristin Lauter, and Andrew Sutherland. Modular polynomials via isogeny volcanoes. *Mathematics of Computation*, 81(278):1201–1231, 2012.

[9] Reinier Bröker and Andrew V Sutherland. An explicit height bound for the classical modular polynomial. *The Ramanujan Journal*, 22:293–313, 2010.

[10] Wouter Castryck and Thomas Decru. Multiradical isogenies. *Arithmetic, Geometry, Cryptography, and Coding Theory*, 779:57–89, 2021.

[11] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.

[12] Craig Costello, David Jao, Patrick Longa, Michael Naehrig, Joost Renes, and David Urbanik. Efficient compression of SIDH public keys. In *EUROCRYPT 2017, Part I 36*, pages 679–706. Springer, 2017.

[13] Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQISignHD: New dimensions in cryptography. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2024.

[14] Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An algorithmic approach to $(2, 2)$-isogenies in the theta model and applications to isogeny-based cryptography. Cryptology ePrint Archive, Paper 2023/1747, 2023.

[15] P. Deligne and M. Rapoport. Les schémas de modules de courbes elliptiques. In *Modular functions of one variable, II (Proc. Internat. Summer School, Univ. Antwerp, Antwerp, 1972)*, pages 143–316. Lecture Notes in Math., Vol. 349, 1973.

[16] R. Dupont. Moyenne arithmetico-geometrique, suites de Borchardt et applications. *These de doctorat, Ecole polytechnique, Palaiseau*, 2006.

[17] Noam D Elkies. Explicit isogenies. *preprint*, 1991.

[18] Andreas Enge. Computing modular polynomials in quasi-linear time. *Mathematics of Computation*, 78(267):1809–1824, 2009.

[19] Robin Hartshorne. *Deformation theory*, volume 257. Springer, 2010.

[20] David Harvey and Joris Van Der Hoeven. Integer multiplication in time o(nlog\,n). *Annals of Mathematics*, 193(2):563–617, 2021.

[21] Henryk Iwaniec and Emmanuel Kowalski. *Analytic number theory*, volume 53. American Mathematical Soc., 2021.

[22] Ernst Kani. The number of curves of genus two with elliptic differentials. *J. reine angew. Math. 485*, pages 93–121, 1997.

[23] Jean Kieffer. Evaluating modular polynomials in genus 2. arXiv preprint arXiv:2010.10094, 2020.

[24] Jean Kieffer. Sign choices in the AGM for genus two theta constants. *Publications mathématiques de Besançon. Algèbre et théorie des nombres*, pages 37–58, 2022.

[25] Jean Kieffer. Certified newton schemes for the evaluation of low-genus theta functions. *Numerical Algorithms*, 93(2):833–862, 2023.

[26] L. Kronecker. Grundzüge einer arithmetischen Theorie der algebraischen Grössen. (Abdruck einer Festschrift zu Herrn E. E. Kummers Doctor-Jubiläum, 10. September 1881.). *Journal für die reine und angewandte Mathematik*, 92:1–122, 1882.

[27] Sabrina Kunzweiler. Efficient computation of $(2^n, 2^n)$-isogenies. *Designs, Codes and Cryptography*, 92(6):1761–1802, 2024.

[28] Sabrina Kunzweiler and Damien Robert. Computing modular polynomials by deformation. `https://github.com/sabrinakunzweiler/modular-polynomials`, 2024.

[29] Antonin Leroux. Computation of Hilbert class polynomials and modular polynomials from supersingular elliptic curves. arXiv preprint arXiv:2301.08531, 2023.

[30] David Lubicz and Damien Robert. Fast change of level and applications to isogenies. *Research in Number Theory (ANTS XV Conference)*, 9(1), 2022.

[31] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In *EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 448–471. Springer, 2023.

[32] Enea Milio. A quasi-linear time algorithm for computing modular polynomials in dimension 2. *LMS Journal of Computation and Mathematics*, 18(1):603–632, 2015.

[33] David Mumford. On the equations defining abelian varieties. II. *Invent. Math.*, 3:75–135, 1967.

[34] Frans Oort. Finite group schemes, local moduli for abelian varieties, and lifting problems. *Compositio Mathematica*, 23(3):265–296, 1971.

[35] Aurel Page and Damien Robert. Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Paper 2023/1766, 2023.

[36] Damien Robert. *Efficient algorithms for abelian varieties and their moduli spaces*. Habilitation à diriger des recherches, Université de Bordeaux (UB), 2021.

[37] Damien Robert. Evaluating isogenies in polylogarithmic time. Cryptology ePrint Archive, Paper 2022/1068, 2022.

[38] Damien Robert. Some applications of higher dimensional isogenies to elliptic curves (overview of results). Cryptology ePrint Archive, Paper 2022/1704, 2022.

[39] Damien Robert. Breaking SIDH in polynomial time. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 472–503. Springer, 2023.

[40] René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.

[41] Éric Schost. Computing parametric geometric resolutions. *Applicable Algebra in Engineering, Communication and Computing*, 13(5):349–393, 2003.

[42] Edoardo Sernesi. *Deformations of algebraic schemes*, volume 334. Springer Science & Business Media, 2007.

[43] The Stacks Project Authors. *Stacks Project*. `https://stacks.math.columbia.edu`, 2018.

[44] John T Tate. p-divisible groups. In *Proceedings of a Conference on Local Fields: NUFFIC Summer School held at Driebergen (The Netherlands) in 1966*, pages 158–183. Springer, 1967.

[45] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.0)*, 2024. `https://www.sagemath.org`.

[46] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences*, 273:238–241, 1971.

[47] Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1100–1111. IEEE, 2022.

Appendix A. Computing $\varphi_\ell \in \mathbb{F}_p[x, y]$, when $\ell \equiv 1 \pmod 4$

Here, we present Algorithm 8, a variant of Algorithm 6, for computing the modular polynomial φ_ℓ over $\mathbb{F}_p$ for $p \in \mathcal{P}_\ell$ and $\ell \equiv 1 \pmod 4$. These modifications were outlined in Subsection 4.3. In particular, we obtain the following explicit version of Theorem 4.5.

Theorem A.1. *On input a prime $\ell \equiv 1 \pmod 4$ and a prime $p \in \mathcal{P}_\ell$, Algorithm 8 returns the modular polynomial φ_ℓ over $\mathbb{F}_p$. The algorithm runs in time $O\left(\log(p) + \ell^2 \log^2(\ell) + n\,\ell^2 \log(\ell)\right)$ over $\mathbb{F}_{p^2}$.*

Proof. This coincides with the proof of Theorem 4.5. □

Inria Bordeaux, Institut de Mathématiques de Bordeaux

Algorithm 8 `modular_polynomial_modp` $\ell \equiv 1 \pmod 4$

Input: A prime $\ell \equiv 1 \pmod 4$, and a prime $p \in \mathcal{P}_\ell$.
Output: The modular polynomial $\varphi_\ell(X,Y) \in \mathbb{F}_p[X,Y]$.

```
    /*Setup*/
1:  Set $E : y^2 = x^3 + 6x^2 + x$ over $\mathbb{F}_{p^2}$
2:  $\iota \leftarrow \iota \in End(E)$ with $\iota \circ \iota = [-4]$
3:  $\gamma \leftarrow [a] + [b]\iota$, where $2^n - 3 \cdot \ell = a^2 + 4b^2$ for some $n$.
4:  Sample $P_3 \in E[3] \setminus \{\mathcal{O}\}$
5:  $C_0 \leftarrow E/\langle P_3 \rangle$, $C_1 \leftarrow E/\langle \gamma(P_3) \rangle$
6:  $\gamma_0, g, g' \leftarrow$ isogenies with $\gamma_0 \circ g = g' \circ \gamma$ as in Eq. 9
7:  Compute $P_{2^n}, Q_{2^n}$ with $C_1[2^n] = \langle P_{2^n}, Q_{2^n} \rangle$
8:  Compute $P_\ell, Q_\ell$ with $E[\ell] = \langle P_\ell, Q_\ell \rangle$
9:  $\tilde{j} \leftarrow j(E) + \epsilon \in \mathbb{F}_{p^2}[\epsilon]/(\epsilon^{\ell+2})$
10: $\mathcal{E} \leftarrow$ elliptic curve with $j(\mathcal{E}) = \tilde{j}$
11: $\mathcal{C}_0 \leftarrow$ codomain of the lift $\tilde{g} : \mathcal{E} \rightarrow \mathcal{C}_0$
    /*computing and lifting all $\ell$-isogenies*/
12: for $k \leftarrow 0, \ldots, \ell$ do
13:     if $k = \ell$ then
14:         $P_k \leftarrow Q_\ell$
15:     else
16:         $P_k \leftarrow P_\ell + k \cdot Q_\ell$
17:     end if
18:     $E_k \leftarrow E/\langle P_k \rangle$ with $f_k : E \rightarrow E_k$
        /*constructing the $(3\ell, 2^n - 3\ell)$-isogeny diamond*/
19:     $E'_k \leftarrow E/\langle \gamma(P_k) \rangle$ with $f'_k : E \rightarrow E'_k$
20:     $K \leftarrow \langle (\hat{\gamma_0}(P_{2^n}), f'_k \circ \hat{g}'(P_{2^n})), (\hat{\gamma_0}(Q_{2^n}), f'_k \circ \hat{g}'(Q_{2^n})) \rangle$
21:     $(\mathcal{C}_0, \mathcal{E}_k, \mathcal{C}_1, \mathcal{E}'_k) \leftarrow$ lift_isogeny_diamond$(C_0, E_k, C_1, E'_k, K, \mathcal{C}_0)$
22:     $\tilde{j}_k \leftarrow j(\mathcal{E}_k)$
23: end for
    /*final step*/
24: $\varphi \leftarrow \prod_{k=0}^{\ell} (Y - \tilde{j}_k)(\epsilon = X - j(E)) \in \mathbb{F}_p[X,Y]$
25: return $\varphi$
```