THE OPEN BOOK SERIES 1

ANTS X
Proceedings of the Tenth
Algorithmic Number Theory Symposium

On the evaluation of modular polynomials

Andrew V. Sutherland

msp

THE OPEN BOOK SERIES **1** (2013)
Tenth Algorithmic Number Theory Symposium
dx.doi.org/10.2140/obs.2013.1.531

On the evaluation of modular polynomials

Andrew V. Sutherland

We present two algorithms that, given a prime ℓ and an elliptic curve $E/\mathbb{F}_q$, directly compute the polynomial $\Phi_\ell(j(E), Y) \in \mathbb{F}_q[Y]$ whose roots are the j-invariants of the elliptic curves that are ℓ-isogenous to E. We do not assume that the modular polynomial $\Phi_\ell(X, Y)$ is given. The algorithms may be adapted to handle other types of modular polynomials, and we consider applications to point counting and the computation of endomorphism rings. We demonstrate the practical efficiency of the algorithms by setting a new point-counting record, modulo a prime q with more than 5,000 decimal digits, and by evaluating a modular polynomial of level $\ell = 100{,}019$.

1. Introduction

Isogenies play a crucial role in the theory and application of elliptic curves. A standard method for identifying (and computing) isogenies uses the classical modular polynomial $\Phi_\ell \in \mathbb{Z}[X, Y]$, which parametrizes pairs of ℓ-isogenous elliptic curves in terms of their j-invariants. More precisely, over a field $\mathbb{F}$ of characteristic not equal to ℓ, the modular equation $\Phi_\ell(j_1, j_2) = 0$ holds if and only if j_1 and j_2 are the j-invariants of elliptic curves defined over $\mathbb{F}$ that are related by a cyclic isogeny of degree ℓ. In practical applications, $\mathbb{F}$ is typically a finite field $\mathbb{F}_q$, and ℓ is a prime, as we shall assume throughout. For the sake of simplicity we assume that q is prime, but this is not essential.

A typical scenario is the following: We are given an elliptic curve $E/\mathbb{F}_q$ and wish to determine whether E admits an ℓ-isogeny defined over $\mathbb{F}_q$, and if so, to identify one or all of the elliptic curves that are ℓ-isogenous to E. This can be achieved by computing the instantiated modular polynomial

$$\phi_\ell(Y) = \Phi_\ell(j(E), Y) \in \mathbb{F}_q[Y],$$

MSC2010: primary 11Y16; secondary 11G15, 11G20.
Keywords: elliptic curves, isogenies, point counting, SEA algorithm.

and finding its roots in $\mathbb{F}_q$ (if any). Each root is the j-invariant of an elliptic curve that is ℓ-isogenous to E over $\mathbb{F}_q$, and every such j-invariant is a root of $\phi_\ell(Y)$.

For large ℓ the main obstacle to obtaining ϕ_ℓ is the size of Φ_ℓ, which is $O(\ell^3 \log \ell)$ bits; storing Φ_ℓ requires several gigabytes for $\ell \approx 10^3$, and many terabytes for $\ell \approx 10^4$ — see [8, Table 1]. In practice, alternative modular polynomials that are smaller than Φ_ℓ by a large constant factor are often used, but their size grows at the same rate, and this quickly becomes the limiting factor, as noted in [15, §5.2] and elsewhere. The 2009 INRIA Project-Team TANC report states:

> "...computing modular polynomials remains the stumbling block for new point counting records. Clearly, to circumvent the memory problems, one would need an algorithm that directly obtains the polynomial specialized in one variable." [26, p. 9]

Here we present just such an algorithm (two in fact), based on the isogeny volcano approach of [8]. Our basic strategy is to compute the instantiated modular polynomial $\phi(Y) = \Phi_\ell(j(E), Y)$ modulo many "suitable" primes p and apply the explicit Chinese remainder theorem modulo q (see Section 2.4 and Section 2.5 for a discussion of the explicit CRT and suitable primes). However, two key issues arise.

First, if we simply lift the j-invariant $j(E)$ from $\mathbb{F}_q \simeq \mathbb{Z}/q\mathbb{Z}$ to $\mathbb{Z}$ and reduce the result modulo p, when we instantiate $\Phi_\ell(j(E), Y)$ the powers of $j(E)$ we compute may correspond to integers that are much larger than the coefficients of Φ_ℓ, forcing us to use many more CRT primes than we would otherwise need. We address this issue by instead exponentiating in $\mathbb{F}_q$, lifting the powers to $\mathbb{Z}$, and then reducing them modulo p. This yields our first algorithm, which is well-suited to situations where q is much larger than ℓ, say $\log q \approx \ell$, as in point-counting applications.

Second, to achieve the optimal space complexity we must avoid computing $\Phi_\ell \bmod p$. Indeed, if $\log q \approx \log \ell$, then $\Phi_\ell \bmod p$ will not be much smaller than $\Phi_\ell \bmod q$. Our second algorithm uses an online approach to avoid storing all the coefficients of $\Phi_\ell \bmod p$ simultaneously. This algorithm is well-suited to situations where $\log q$ is not dramatically larger than $\log \ell$, say $O(\log \ell)$ or $O(\log^2 \ell)$. This occurs, for example, in algorithms that compute the endomorphism ring of an elliptic curve [3], or algorithms to evaluate isogenies of large degree [27].

Under the generalized Riemann hypothesis (GRH), our first algorithm has an expected running time of $O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$ and uses $O(\ell^2 \log \ell + \ell \log q)$ space, assuming $\log q = O(\ell \log \ell)$.[1] This time complexity is the same as (and in practice is faster than) the time to compute Φ_ℓ, and the space complexity is reduced by up to a factor of ℓ. When $\log q \approx \ell$ the space complexity is nearly optimal: quasilinear

[1] See Theorem 4 for a more precise bound. Throughout, we write $\operatorname{llog} n$ for $\log \log n$ and $\operatorname{lllog} n$ for $\log \log \log n$.

in the size of ϕ_ℓ. The second algorithm uses $O(\ell^3(\log q + \log \ell)\log^{1+o(1)}\ell)$ time and $O(\ell \log q + \ell \log \ell)$ space, under the GRH. Its space complexity is optimal for $q = \Omega(\ell)$, and when $\log q = O(\log^{2-\epsilon}\ell)$ its time complexity is better than the time to compute Φ_ℓ. For $\log q \gg \log^2\ell$ its running time becomes less attractive and the first algorithm may be preferred; alternatively, see Section 3.4 for a hybrid approach.

In conjunction with the SEA algorithm, the first algorithm allows us to compute the cardinality of an elliptic curve modulo a prime q with a heuristic[2] running time of $O(n^4 \log^3 n \operatorname{llog} n)$, using $O(n^2 \log n)$ space, where $n = \log q$. To our knowledge, all alternative approaches applicable to prime fields increase at least one of these bounds by a factor of n or more. The running time is competitive with SEA implementations that rely on precomputed modular polynomials (as can be found in Magma [4] and PARI [32]), and can easily handle much larger values of q.

As an important practical optimization, we also evaluate modular polynomials $\phi_\ell^f(Y) = \Phi_\ell^f(f(E), Y)$ defined by modular functions $f(z)$ other than the j-function. This includes the Weber $\mathfrak{f}$-function, whose modular polynomials are smaller than the classical modular polynomial by a factor of 1728 and can be computed much more quickly (by roughly the same factor). This speedup also applies when computing ϕ_ℓ^f.

To demonstrate the capability of the new algorithms, we use a modified version of the SEA algorithm to count points on an elliptic curve modulo a prime of more than 5,000 decimal digits, and evaluate a modular polynomial of level $\ell = 100{,}019$ modulo a prime of more than 25,000 decimal digits.

2. Background

This section contains a brief summary of background material that can be found in standard references such as [31; 39; 40], or in the papers [8; 42], both of which exploit isogeny volcanoes using a CRT-based approach, as we do here. For the sake of brevity, we recall only the results we need, and only in the generality necessary.

To simplify the presentation, we assume throughout that $\mathbb{F}_p$ and $\mathbb{F}_q$ denote prime fields with $\ell \neq p, q$, and, where relevant, that q is sufficiently large (typically $q > 2\ell$). But this assumption is not needed for our main result; Algorithms 1 and 2 work correctly for any prime q (even $q = \ell$), and can be extended to handle nonprime q.

2.1. ***Isogenies.*** Let E be an elliptic curve defined over a field $\mathbb{F}$. Recall that an *isogeny* is a nonconstant morphism $\psi : E \to \widetilde{E}$ of elliptic curves that is also a group

[2] The heuristic relates to the distribution of Elkies primes and is a standard assumption made when using the SEA algorithm; without it there is no advantage over Schoof's algorithm.

homomorphism from $E(\overline{\mathbb{F}})$ to $\widetilde{E}(\overline{\mathbb{F}})$. The kernel of an isogeny is a finite subgroup of $E(\overline{\mathbb{F}})$, and when ψ is separable, the size of its kernel is equal to its degree. Conversely, every finite subgroup G of $E(\overline{\mathbb{F}})$ is the kernel of a separable isogeny (defined over the fixed field of the stabilizer of G in $\operatorname{Gal}(\overline{\mathbb{F}}/\mathbb{F})$). We say that ψ is *cyclic* if its kernel is cyclic, and call ψ an *N-isogeny* when it has degree N. Note that an isogeny of prime degree $\ell \neq \operatorname{char}\mathbb{F}$ is necessarily cyclic and separable.

The classical modular polynomial Φ_N is the minimal polynomial of the function $j(Nz)$ over the field $\mathbb{C}(j)$, where $j(z)$ is the modular j-function. As a polynomial in two variables, $\Phi_N \in \mathbb{Z}[X, Y]$ is symmetric in X and Y and has the defining property that the roots of $\Phi_\ell(j(E), Y)$ are precisely the j-invariants of the elliptic curves $\widetilde{E}$ that are related to E by a cyclic N-isogeny. In this paper $N = \ell$ is prime, in which case $\Phi_\ell(X, Y)$ has degree $\ell + 1$ in each variable.

If E is given by a short Weierstrass equation $Y^2 = X^3 + a_4X + a_6$, then ψ can be expressed in the form

$$\psi(x, y) = \left(\psi_1(x),\ cy\frac{d}{dx}\psi_1(x)\right)$$

for some $c \in \overline{\mathbb{F}}^*$. When $c = 1$ we say that ψ and its image are *normalized*. Given a finite subgroup G of $E(\overline{\mathbb{F}})$, a normalized isogeny with G as its kernel can be constructed using Vélu's formulae [45], along with an explicit equation for its image $\widetilde{E}$. Conversely, suppose we are given a root $\tilde{\jmath} = j(\widetilde{E})$ of $\phi_\ell(Y) = \Phi_\ell(j(E), Y)$, and also the values of $\Phi_X(j, \tilde{\jmath})$, $\Phi_Y(j, \tilde{\jmath})$, $\Phi_{XX}(j, \tilde{\jmath})$, $\Phi_{XY}(j, \tilde{\jmath})$, and $\Phi_{YY}(j, \tilde{\jmath})$, where $j = j(E)$ and

$$\Phi_X = \frac{\partial}{\partial X}\Phi_\ell, \qquad \Phi_Y = \frac{\partial}{\partial Y}\Phi_\ell,$$
$$\Phi_{XX} = \frac{\partial^2}{\partial X^2}\Phi_\ell, \quad \Phi_{XY} = \frac{\partial^2}{\partial X\partial Y}\Phi_\ell, \quad \Phi_{YY} = \frac{\partial^2}{\partial Y^2}\Phi_\ell.$$

To this data we may apply an algorithm of Elkies [13] that computes an equation for $\widetilde{E}$ that is the image of a normalized ℓ-isogeny $\psi : E \to \widetilde{E}$, along with an explicit description of its kernel: the monic polynomial $h_\ell(X)$ whose roots are the abscissae of the nontrivial points in $\ker\psi$; see [19, Algorithm 27]. The quantities $\Phi_{XX}(j, \tilde{\jmath})$, $\Phi_{XY}(j, \tilde{\jmath})$, and $\Phi_{YY}(j, \tilde{\jmath})$ are not strictly necessary; the equation for $\widetilde{E}$ depends only on j, $\tilde{\jmath}$, $\Phi_X(j, \tilde{\jmath})$ and $\Phi_Y(j, \tilde{\jmath})$, and we may then apply algorithms of Bostan et al. [5] to compute $h_\ell(X)$, and an equation for ψ, directly from E and $\widetilde{E}$.

2.2. ***Explicit CM theory.*** Recall that the endomorphism ring of an ordinary elliptic curve E over a finite field $\mathbb{F}_p$ is isomorphic to an order $\mathcal{O}$ in an imaginary quadratic field K. In this situation E is said to have *complex multiplication* (CM) by $\mathcal{O}$. The elliptic curve $E/\mathbb{F}_p$ is the reduction of an elliptic curve $\widehat{E}/\mathbb{C}$ that also has

CM by $\mathcal{O}$. The j-invariant of $\widehat{E}$ generates the ring class field $K_{\mathcal{O}}$ of $\mathcal{O}$, and its minimal polynomial over K is the *Hilbert class polynomial* $H_{\mathcal{O}} \in \mathbb{Z}[X]$, whose degree is the class number $h(\mathcal{O})$.[3] The prime p splits completely in $K_{\mathcal{O}}$, and $H_{\mathcal{O}}$ splits completely in $\mathbb{F}_p[X]$. For $p > 3$, the prime p splits completely in $K_{\mathcal{O}}$ if and only if it satisfies the norm equation $4p = t^2 - v^2 D$, where $D = \operatorname{disc} \mathcal{O}$, and for $D < -4$ the integers $t = t(p)$ and $v = v(p)$ are uniquely determined up to sign.

We define the set

$$\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p) = \{ j(E) : E/\mathbb{F}_p \text{ with } \operatorname{End}(E) \simeq \mathcal{O} \},$$

which consists of the roots of $H_{\mathcal{O}}$ in $\mathbb{F}_p$. Let $\iota : \mathcal{O} \hookrightarrow \operatorname{End}(E)$ denote the normalized embedding (so $\iota(\alpha)^* \omega = \alpha\omega$ for all $\alpha \in \mathcal{O}$ and invariant differentials ω on E; see [40, Proposition II.1.1, p. 97]). The ideals of $\mathcal{O}$ act on $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ via isogenies as follows. Let $\mathfrak{a}$ be an $\mathcal{O}$-ideal of norm N, and define $E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} \ker \iota(\alpha)$. There is a separable N-isogeny from E to $\widetilde{E} = E/E[\mathfrak{a}]$, and the action of $\mathfrak{a}$ sends $j(E)$ to $j(\widetilde{E})$. Principal ideals act trivially, and this induces a regular action of the class group $\operatorname{Cl}(\mathcal{O})$ on $\operatorname{Ell}_O(\mathbb{F}_p)$. Thus $\operatorname{Ell}_O(\mathbb{F}_p)$ is a principal homogeneous space, a *torsor*, for $\operatorname{Cl}(\mathcal{O})$.

Writing the $\operatorname{Cl}(\mathcal{O})$-action on the left, we note that if $\mathfrak{a}$ has prime norm ℓ, then $\Phi_\ell(j, [\mathfrak{a}]j) = 0$ for all $j \in \operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$. For ℓ not dividing $v(p)$, the polynomial $\phi_\ell(Y) = \Phi_\ell(j, Y)$ has either one or two roots in $\mathbb{F}_p$, depending on whether ℓ ramifies or splits in K. In the latter case, the two roots $[\mathfrak{a}]j$ and $[\mathfrak{a}^{-1}]j$ can be distinguished using the Elkies kernel polynomial $h_\ell(X)$, as described in [6, §5] and [20, §3].

2.3. ***Polycyclic presentations.*** In order to efficiently realize the action of $\operatorname{Cl}(\mathcal{O})$ on $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$, it is essential to represent elements of $\operatorname{Cl}(\mathcal{O})$ in terms of a set of generators with small norm. We will choose $\mathcal{O}$ so that $\operatorname{Cl}(\mathcal{O})$ is generated by ideals of norm bounded by $O(1)$, via [8, Theorem 3.3], but these generators will typically not be independent. Thus, as explained in [42, §5.3], we use polycyclic presentations.

Any sequence of generators $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_k)$ for a finite abelian group G defines a polycyclic series

$$1 = G_0 \lhd G_1 \lhd \cdots \lhd G_{k-1} \lhd G_k = G,$$

with $G_i = \langle \alpha_1, \dots, \alpha_i \rangle$, in which every quotient $G_i/G_{i-1} \simeq \langle \alpha_i \rangle$ is cyclic. We associate to $\boldsymbol{\alpha}$ the sequence of *relative orders* $r(\boldsymbol{\alpha}) = (r_1, \dots, r_k)$ defined by $r_i = [G_i : G_{i-1}]$. Every element $\beta \in G$ has a unique $\boldsymbol{\alpha}$*-representation* of the form

$$\beta = \boldsymbol{\alpha}^{\boldsymbol{e}} = \alpha_1^{e_1} \cdots \alpha_k^{e_k} \quad (0 \le e_i < r_i).$$

[3] As in [1], we call $H_{\mathcal{O}}$ a Hilbert class polynomial even when $\mathcal{O}$ is not the maximal order.

We also associate to $\boldsymbol{\alpha}$ the matrix of power relations $s(\boldsymbol{\alpha}) = [s_{ij}]$ defined by

$$\alpha_i^{r_i} = \alpha_1^{s_{i,1}} \alpha_2^{s_{i,2}} \cdots \alpha_{i-1}^{s_{i,i-1}} \quad (0 \le s_{ij} < r_j),$$

with $s_{ij} = 0$ for $i \le j$.

We call $\boldsymbol{\alpha}$, together with $r(\boldsymbol{\alpha})$ and $s(\boldsymbol{\alpha})$, a (*polycyclic*) *presentation* for G, and if all the r_i are greater than 1, we say that the presentation is *minimal*. A generic algorithm to compute a minimal polycyclic presentation is given in [42, Algorithm 2.2]. Having constructed such an $\boldsymbol{\alpha}$, we can efficiently enumerate $G = \mathrm{Cl}(\mathcal{O})$ (or the torsor $\mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_q)$, given a starting point), by enumerating $\boldsymbol{\alpha}$-representations.

2.4. ***Explicit CRT.*** Let $p_1, \dots, p_n$ be primes with product M, let $M_i = M/p_i$, and let $a_i M_i \equiv 1 \bmod p_i$. If $c \in \mathbb{Z}$ satisfies $c \equiv c_i \bmod p_i$, then $c \equiv \sum_i c_i a_i M_i \bmod M$. If $M > 2|c|$, this congruence uniquely determines c. This is the usual CRT method.

Now suppose $M > 4|c|$ and let q be a prime (or any integer). Then we may apply the *explicit CRT mod q* [2, Theorem 3.1] to compute

$$c \equiv \left(\sum_i c_i a_i M_i - rM \right) \bmod q, \tag{1}$$

where r is the closest integer to $\sum_i c_i a_i / p_i$; when computing r, it suffices to approximate each $c_i a_i / p_i$ to within $1/(4n)$, by [2, Theorem 2.2].

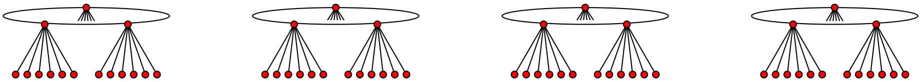
As described in [42, §6], we may use the explicit CRT to simultaneously compute $c \bmod q$ for many integers c (the coefficients of ϕ_ℓ, for example), using an *online algorithm*. We first precompute the a_i and $a_i M_i \bmod q$. Then, for each prime p_i, we determine the values c_i for all the coefficients c (by computing $\phi_\ell \bmod p_i$), update two partial sums for each coefficient, one for $\sum c_i a_i M_i \bmod q$ and one for $\sum c_i a_i / p_i$, and then discard the c_i. When the computations for all the p_i have been completed (these may be performed in parallel), we compute r and apply (1) for each coefficient. The space required by the partial sums is just $O(\log q)$ bits per coefficient. See [42, §6] for further details, including algorithms for each step.

2.5. ***Modular polynomials via isogeny volcanoes.*** For distinct primes ℓ and p, we define the *graph of ℓ-isogenies* $\Gamma_\ell(\mathbb{F}_p)$, with vertex set $\mathbb{F}_p$ and edges (j_1, j_2) present if and only if $\Phi_\ell(j_1, j_2) = 0$. Ignoring the connected components of 0 and 1728, the ordinary components of $\Gamma_\ell(\mathbb{F}_p)$ are *ℓ-volcanoes* [18; 30], a term we take to include cycles as a special case [42]. In this paper we focus on ℓ-volcanoes of a particular form, for which we can compute $\Phi_\ell \bmod p$ very quickly, via [8, Algorithm 2.1].

Let $\mathcal{O}$ be an order in an imaginary quadratic field K with maximal order $\mathcal{O}_K$, let ℓ be an odd prime not dividing $[\mathcal{O}_K : \mathcal{O}]$, let $\mathcal{O}'$ be the order of index l in $\mathcal{O}$, and assume

$D = \operatorname{disc}\mathcal{O} < -4$. Let p be a prime of the form $4p = t^2 - \ell^2 v^2 D$ with $\ell \nmid v$ and $p \equiv 1 \bmod \ell$. Then p splits completely in the ring class fields of $\mathcal{O}$ and $\mathcal{O}'$, but not in the ring class field of the order of index ℓ^2 in $\mathcal{O}$. The requirement $p \equiv 1 \bmod \ell$ ensures that for $j(E) \in \mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ we can choose E so that $E[\ell] \subset E(\mathbb{F}_p)$, which is critical to the efficiency of both the algorithm in [8] and our algorithms here.

The components of $\Gamma_\ell(\mathbb{F}_p)$ that intersect $\mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ are isomorphic ℓ-volcanoes with two levels: the *surface*, whose vertices lie in $\mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$, and the *floor*, whose vertices lie in $\mathrm{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$. Each vertex on the surface is connected to $1 + \left(\frac{D}{\ell}\right) = 0, 1$ or 2 *siblings* on the surface, and $\ell - \left(\frac{D}{\ell}\right)$ *children* on the floor. An example with $\ell = 7$ and $\left(\frac{D}{\ell}\right) = 1$ is shown below:



Provided that $h(\mathcal{O}) \geq \ell + 2$, this set of ℓ-volcanoes contains enough information to completely determine $\Phi_\ell \bmod p$. This is the basis of the algorithm in [8, Algorithm 2.1], which we adapt here. Selecting a sufficiently large set of such primes p allows one to compute Φ_ℓ over $\mathbb{Z}$ (via the CRT), or modulo an arbitrary prime q (via the explicit CRT). In order to achieve the best complexity bounds, it is important to choose both the order $\mathcal{O}$ and the primes p carefully. We thus introduce the following definitions, in which c_1 and c_2 are fixed constants that do not depend on ℓ or $\mathcal{O}$. (In our implementation we used $c_1 = 1.5$ and $c_2 = 256$.)

Definition 1. Let $\mathcal{O}$ be a quadratic order with discriminant $D = u^2 D_0 < 0$, with D_0 fundamental, and let $c_1, c_2 > 1$ be constants. We say that $\mathcal{O}$ is *suitable* for ℓ if

(1) $\ell + 2 \leq h(\mathcal{O}) \leq c_1 \ell$,

(2) $4 < |D_0| \leq c_2^2$,

(3) $\ell^2 \leq |D| \leq c_2^2 \ell^2$,

(4) $\gcd(u, 2\ell D) = 1$, and

(5) $l_0 < \min(c_2, \ell)$ for all primes $l_0 \mid u$.

This definition combines the criteria in [8, Definition 4.2] and [8, Theorem 5.1]. Provided that c_1 and c_2 are not too small, suitable orders exist for every odd prime ℓ; with $c_1 = 4$ and $c_2 = 16$, for example, we may use orders with $D = -7 \cdot 3^{2n}$ for all $\ell > 3$. Ideally we want c_1 to be as close to 1 as possible, but this makes it harder to find suitable orders. For the asymptotic analysis, any values of c_1 and c_2 will do.

Definition 2. A prime p is *suitable* for ℓ and $\mathcal{O}$ if $p \equiv 1 \bmod \ell$ and p satisfies $4p = t^2 - \ell^2 v^2 D$ for some $t, v \in \mathbb{Z}$ with $\ell \nmid v$ and $\omega(v) \leq 2\log(\log v + 3)$.

The function $\omega(v)$ counts the distinct prime divisors of v. The bound on $\omega(v)$ ensures that if $\mathcal{O}$ is suitable for ℓ then many small primes split in $\mathcal{O}$ and do not divide u or v. Such primes allow us to more efficiently enumerate $\mathrm{Cl}(\mathcal{O})$ and $\mathrm{Cl}(\mathcal{O}')$.

2.6. ***Selecting primes with the GRH.*** In order to apply the isogeny volcano method to compute $\Phi_\ell \bmod q$ (or $\phi_\ell \bmod q$, as we shall do), we need a sufficiently large set S of suitable primes p. We deem S to be sufficiently large whenever

$$\sum_{p\in S} \log p \geq B + \log 4,$$

where B is an upper bound on the logarithmic height of the integers whose reductions mod q we wish to compute with the explicit CRT. For $\Phi_\ell(X,Y) = \sum_{i,j} a_{ij}X^iY^j$, we may bound $h(\Phi_\ell) = \log \max_{i,j} |a_{ij}|$ using

$$h(\Phi_\ell) \leq 6\ell \log \ell + 18\ell, \tag{2}$$

$$h(\Phi_\ell) \leq 6\ell \log \ell + 16\ell + 14\sqrt{\ell}\log \ell, \tag{3}$$

as proved in [9]. (We prefer the latter bound when $\ell > 3187$.)

Heuristically (and in practice), it is easy to construct the set S. Given an order $\mathbb{O}$ of discriminant D suitable for ℓ, we fix $v=2$ if $D \equiv 1 \bmod 8$ and $v=1$ otherwise, and for increasing $t \equiv 2 \bmod \ell$ of correct parity we test whether $p=(t^2-v^2\ell^2D)/4$ is prime. We add each prime value of p to S, and stop when S is sufficiently large.

Unfortunately, we cannot prove that this method will find *any* primes, even under the GRH. Instead, we use Algorithm 6.2 in [8], which picks an upper bound x and generates random integers t and v in suitable intervals to obtain candidate primes $p = (t^2 - v^2\ell^2D)/4 \leq x$ that are then tested for primality. The algorithm periodically increases x, so its expected running time is $O(B^{1+\epsilon})$, even without the GRH. To ensure that the bound on $\omega(v)$ in Definition 2 is satisfied, unsuitable v's are discarded; this occurs with negligible probability.

Under the GRH, there are effective constants $c_3, c_4 > 0$ such that $x \geq c_3\ell^6 \log^4 \ell$ guarantees at least $c_4\ell^3 \log^3 \ell$ suitable primes less than x, by [8, Theorem 4.4]. Asymptotically, this is far more than the $O(\ell)$ primes we need to compute $\Phi_\ell \bmod q$. Here we may consider larger values of B, and in general, $x = O(B^2 + \ell^6 \log^4 \ell)$ suffices. We note that S contains $O(B/\log B)$ primes (unconditionally), and under the GRH we have $\log p = O(\log B + \log \ell)$ for all $p \in S$.

3. Algorithms

Let q be a prime and let E be an elliptic curve over $\mathbb{F}_q$. A simple algorithm to compute $\phi_\ell(Y) = \Phi_\ell(j(E), Y) \in \mathbb{F}_q[Y]$ with the explicit CRT works as follows. Let $\hat{\jmath}$ be the integer in $[0, q-1]$ corresponding to $j(E) \in \mathbb{F}_q \simeq \mathbb{Z}/q\mathbb{Z}$. For a sufficiently large set S of suitable primes p, compute $\Phi_\ell(X,Y) \bmod p$ using the isogeny volcano algorithm and evaluate $\Phi_\ell(\hat{\jmath}, Y) \bmod p$ to obtain $\bar{\phi}_\ell \in \mathbb{F}_p[Y]$, and use the explicit CRT mod q to eventually obtain $\phi_\ell \in \mathbb{F}_q[Y]$.

This naïve algorithm suffers from two significant defects. The most serious is that we may now require a much larger set S than is needed to compute $\Phi_\ell \bmod q$.

Compared to the coefficients of Φ_ℓ, which have height $h(\Phi_\ell) = O(\ell \log \ell)$ bounded by inequalities (2) and (3), we now need to use the $O(\ell \log \ell + \ell \log q)$ bound

$$h(\Phi_\ell(\hat{\jmath}, Y)) \leq h(\Phi_\ell) + (\ell + 1) \log q + \log(\ell + 2), \tag{4}$$

since $\Phi_\ell(\hat{\jmath}, Y)$ involves powers of $\hat{\jmath}$ up to $\hat{\jmath}^{\ell+1}$.

If $\log q$ is comparable to $\log \ell$, then the difference between the bounds in inequalities (2) and (3) and the bound in inequality (4) may be negligible. But when $\log q$ is comparable to ℓ, using the bound in inequality (4) increases the running time dramatically. This issue is addressed by Algorithm 1.

The second defect of the naïve algorithm is that although its space complexity may be significantly better than the $O(\ell^2 \log q)$ space required to compute Φ_ℓ mod q, it is still quasiquadratic in ℓ. But the size of ϕ_ℓ is linear in ℓ, so we might hope to do better, and indeed we can. This is achieved by Algorithm 2.

A hybrid approach that combines aspects of both algorithms is discussed in Section 3.4.

3.1. ***Algorithm 1.*** The increase in the height bound from inequalities (2) and (3) to inequality (4) is caused by the fact that *we are exponentiating in the wrong ring*. Rather than lifting $j(E) \in \mathbb{F}_q$ to the integer $\hat{\jmath}$ and computing powers of its reduction in $\mathbb{F}_p$ (which simulates powering in $\mathbb{Z}$), we should instead compute powers $j(E)$, $j(E)^2, \ldots, j(E)^{\ell+1}$ in $\mathbb{F}_q$, lift these values to integers $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{\ell+1}$, and work with their reductions in $\mathbb{F}_p$, as in [43, §4.4] (a similar strategy is used in [28]). Of course the reductions of $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{\ell+1}$ need not correspond to powers of any particular element in $\mathbb{F}_p$; nevertheless, if we simply replace each occurrence of X^i in the modular polynomial $\Phi_\ell(X, Y)$ mod p with $\hat{x}_i$ mod p, we achieve the same end result using a much smaller height bound.

We now present Algorithm 1 to compute $\phi(Y) = \phi_\ell(Y) = \Phi_\ell(j(E), y)$. If desired, the algorithm can also compute the polynomials

$$\phi_X(Y) = \frac{\partial \Phi_\ell}{\partial X}(j(E), Y) \quad \text{and} \quad \phi_{XX}(Y) = \frac{\partial^2 \Phi_\ell}{\partial X^2}(j(E), Y),$$

which may be used to compute normalized isogenies, as described in Section 3.8.

Algorithm 1.

Input: An odd prime ℓ, a prime q, and $j(E) \in \mathbb{F}_q$.

Output: The polynomial $\phi(Y) = \Phi_\ell(j(E), Y) \in \mathbb{F}_q[Y]$, and, optionally, $\phi_X(Y)$ and $\phi_{XX}(Y)$.

1. Select an order $\mathcal{O}$ suitable for ℓ and a set of suitable primes S (see Section 2.6), using the height bound $B = 6\ell \log \ell + 18\ell + \log q + 3 \log(\ell + 2)$.
2. Compute the Hilbert class polynomial $H_{\mathcal{O}}(X)$ via [42, Algorithm 2].

3. Perform CRT precomputation mod q using S (see Section 2.4).
4. Compute integers $\hat{x}_i \in [0, q-1]$ such that $\hat{x}_i \equiv j(E)^i \bmod q$, for $0 \le i \le \ell + 1$.
5. For each prime $p \in S$:
   (a) Compute $\Phi_\ell(X, Y) \bmod p$ using $H_{\mathcal{O}}$, via [8, Algorithm 2.1].
   (b) Compute
   $$\bar{\phi}(Y) = \sum_{i,j} a_{ij} \hat{x}_i Y^j \bmod p,$$
   where $\Phi_\ell(X, Y) = \sum_{i,j} a_{ij} X^i Y^j$.
   (c) (Optional.) Compute
   $$\bar{\phi}_X(Y) = \sum_{i,j} i a_{ij} \hat{x}_i Y^j \bmod p$$
   and
   $$\bar{\phi}_{XX}(Y) = \sum_{i,j} i(i-1) a_{ij} \hat{x}_i Y^j \bmod p.$$
   (d) Update CRT sums for each coefficient c_i of $\bar{\phi}$ (and of $\bar{\phi}_X$ and $\bar{\phi}_{XX}$).
6. Perform CRT postcomputation to obtain ϕ (and ϕ_X and ϕ_{XX}) mod q.
7. Output ϕ and (optionally) ϕ_X and ϕ_{XX}.

Proposition 3. *The output $\phi(Y)$ of Algorithm 1 is equal to $\Phi_\ell(j(E), Y)$ (and $\phi_X(Y) = (\partial \Phi_\ell / \partial X)(j(E), Y)$ and $\phi_{XX}(Y) = (\partial^2 \Phi_\ell / \partial X^2)(j(E), Y)$).*

Proof. Let $\varphi = \Phi_\ell(\hat{\jmath}, Y) \in \mathbb{F}_q[Y]$. Let $\hat{x}_i \in \mathbb{Z}$ be as in step 4. Write Φ_ℓ as $\sum_{i,j} a_{ij} X^i Y^j$, with $a_{ij} \in \mathbb{Z}$ and let $\hat{\phi} = \sum_{i,j} a_{ij} \hat{x}_i Y^j \in \mathbb{Z}[Y]$. Then $\varphi \equiv \hat{\phi} \bmod q$, and $\bar{\phi} \equiv \hat{\phi} \bmod p$. To prove $\phi = \varphi$, we only need to show $h(\hat{\phi}) \le B$. We have

$$\left| \sum_i a_{ij} \hat{x}_i \right| \le (\ell + 2) q \exp h(\Phi_\ell),$$

for $0 \le j \le \ell + 1$, hence $h(\hat{\phi}) \le B$. The proofs for ϕ_X and ϕ_{XX} are analogous. We note that the last term in B can be reduced to $\log(\ell + 2)$ if ϕ_X and ϕ_{XX} are not being computed. □

Theorem 4. *Assume the GRH. Then the expected running time of Algorithm 1 is $O(\ell^2 B \log^2 B \operatorname{llog} B)$, where $B = O(\ell \log \ell + \log q)$ is as specified in step 1. The algorithm uses $O(\ell \log q + \ell^2 \log B)$ space.*

Proof. We use $\mathsf{M}(n) = O(n \log n \operatorname{llog} n)$ to denote the cost of multiplication [35]. For step 1, we assume the time spent selecting $\mathcal{O}$ is negligible (as noted in Section 2.5, one may simply choose orders with discriminants of the form $D = -7 \cdot 3^{2n}$), and under the GRH the expected time to construct S is $O(B^{1+\epsilon})$, using $O(B)$ space, as explained in Section 2.6. Step 2 uses $O(\ell^{2+\epsilon})$ expected time and $O(\ell(\log \ell + \log q))$ space, by [42, Theorem 1], since $h(D) = O(\ell)$. An analysis as in [42, §6.3] shows that the total cost of all CRT operations is $O(\ell \mathsf{M}(B) \log B)$ time and $O(\ell \log q)$ space. Step 4 uses $O(\ell \mathsf{M}(\log q))$ time and $O(\ell \log q)$ space.

The set S contains $O(B/\log B)$ primes p, and under the GRH, $\log p = O(\log B)$; see Section 2.6. Step 5(a) dominates the cost per p, taking $O(\ell^2 \log^3 B \operatorname{llog} B)$ expected time and $O(\ell^2 \log B)$ space, by [8]. This yields an $O(\ell^2 B \log^2 B \operatorname{llog} B)$ bound for step 5, which dominates, and the total space is $O(\ell \log q + \ell^2 \log B)$. □

When $\log q = \Theta(\ell)$, the time bound in Theorem 4 reduces to $O(\ell^3 \log^3 \ell \operatorname{llog} \ell)$, the same as the time to compute $\Phi_\ell \bmod q$, and the space bound is $O(\ell \log \ell \log q)$, which is within an $O(\log \ell)$ factor of the best possible.

3.2. *Algorithm 2.*

We now present Algorithm 2, which for $q > \ell$ has optimal space complexity $O(\ell \log q)$. When q is reasonably small, say $\log q = o(\log^2 \ell)$, Algorithm 2 is also faster than Algorithm 1, but when $\log q$ is large it may be much slower, since it uses the same height bound — inequality (4) — as the naïve approach (see Section 3.4 for a hybrid approach). The computation of $\bar{\phi} \in \mathbb{F}_p[Y]$ is more intricate, so we present it separately as Algorithm 2.1. Unlike Algorithm 1, it is not so easy to also compute ϕ_X and ϕ_{XX}, but an alternative method to compute normalized isogenies using Algorithm 2 is given in Section 3.8.

Algorithm 2.

```
Input:   An odd prime $\ell$, a prime $q$, and $j(E) \in \mathbb{F}_q$.
Output:  The polynomial $\phi(Y) = \Phi_\ell(j(E), Y) \in \mathbb{F}_q[Y]$.
1. Select an order $\mathbb{O}$ suitable for $\ell$ and a suitable set of primes $S$ (see Section 2.6),
   using the height bound $B = 6\ell \log \ell + 18\ell + (\ell + 1) \log q + \log(\ell + 2)$.
2. Compute the Hilbert class polynomial $H_{\mathbb{O}}$ via [42, Algorithm 2].
3. Perform precomputation for the explicit CRT mod $q$ using $S$.
4. Let $\hat{\jmath}$ be the integer in $[0, q-1]$ congruent to $j(E) \bmod q$.
5. For each prime $p \in S$:
   (a) Compute $\bar{\phi}(Y) = \Phi_\ell(\hat{\jmath}, Y) \bmod p$ using $\mathbb{O}$ and $H_{\mathbb{O}}$ via Algorithm 2.1.
   (b) Update CRT sums for each coefficient $c_i$ of $\bar{\phi}$.
6. Perform postcomputation for the explicit CRT to obtain $\phi \in \mathbb{F}_q[X]$.
7. Output $\phi$.
```

Proposition 5. *The output $\phi(Y)$ of Algorithm 2 is equal to $\Phi_\ell(j(E), Y)$.*

Proof. This follows immediately from Proposition 7 below and the bound

$$h(\Phi_\ell(\hat{\jmath}, Y)) = \log \max_j \left| \sum_i a_{ij} \hat{\jmath}^i \right| \le \log(\ell + 2) + (\ell + 1) \log q + h(\Phi_\ell) \le B$$

on the height of $\Phi_\ell(\hat{\jmath}, Y) \in \mathbb{Z}[Y]$. □

Theorem 6. *Assume the GRH and that* $\log q = O(\ell^k)$ *for some constant* k. *The expected running time of Algorithm 2 is* $O(\ell^3(\log q + \log\ell)\log\ell\,\operatorname{llog}^2\ell\,\operatorname{lllog}^2\ell)$ *and it uses* $O(\ell\log q + \ell\log\ell)$ *space.*

Proof. As in the proof of Theorem 4, the expected running time is dominated by the time to compute $\bar{\phi}(Y)$, which by Theorem 8 is $O(\ell^2\log^2 p\,\operatorname{llog}^2 p\,\operatorname{lllog}^2 p)$. There are $O(B/\log B)$ primes $p \in S$, and under the GRH we have $\log p = O(\log B) = O(\log\ell)$. The space complexity is dominated by the $O(B) = O(\ell\log\ell + \ell\log q)$ size of S. □

3.3. ***Algorithm 2.1.*** The algorithm in [8, Algorithm 2.1] computes $\Phi_\ell \bmod p$ by enumerating the sets $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ and $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$, where $\mathcal{O}' = \mathbb{Z} + \ell\mathcal{O}$, the latter of which contains approximately ℓ^2 elements. To achieve a space complexity that is quasilinear in ℓ, we cannot afford to store the entire set $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$. We must compute $\Phi_\ell(\hat{\jmath}, Y) \bmod p$ using an online algorithm, processing each $j_k \in \operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ as we enumerate it, and then discarding it. Let us consider how this may be done.

Let $y_1, \dots, y_{h(\mathcal{O})}$ be the elements of $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$, as enumerated using a polycyclic presentation $\boldsymbol{\alpha}$ for $\operatorname{Cl}(\mathcal{O})$. Each y_i is ℓ-isogenous to a set S_i of *siblings* in $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$, and to a set C_i of *children* in $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$; see Section 2.5. Thus we have

$$\Phi_\ell(X, y_i) = \Bigl(\prod_{\tilde{\jmath}\in S_i}(X - \tilde{\jmath})\Bigr)\Bigl(\prod_{\tilde{\jmath}\in C_i}(X - \tilde{\jmath})\Bigr).$$

The siblings can be readily identified in our enumeration of $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ using the CM action (see Section 2.2). To identify the children, we need to be able to determine, for any given $j \in \mathcal{O}'$, the set C_i in which it lies. Each C_i is a subset of the torsor $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ corresponding to a coset of the subgroup $C \subset \operatorname{Cl}(\mathcal{O}')$ generated by the ideals of norm ℓ^2; indeed, two children have the same parent if and only if they are related by an isogeny of degree ℓ^2 (the composition of two ℓ-isogenies).

The group $\operatorname{Cl}(\mathcal{O}')$ acts on the cosets of C, and we need to compute this action explicitly in terms of the polycyclic presentation $\boldsymbol{\beta}$ used to enumerate $\operatorname{Cl}(\mathcal{O}')$. This problem is addressed by a generic group algorithm in the next section that computes a polycyclic presentation $\boldsymbol{\gamma}$ for the quotient $\operatorname{Cl}(\mathcal{O}')/C$, along with the $\boldsymbol{\gamma}$-representation of the image of each generator in $\boldsymbol{\beta}$.

As we enumerate the elements j_k of $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$, starting from a child j_1 of y_1 obtained via Vélu's algorithm, we keep track of the element $\delta_k \in \operatorname{Cl}(\mathcal{O}')$ whose action sends j_1 to j_k. The image of δ_k in $\operatorname{Cl}(\mathcal{O}')/C$ is the coset of C corresponding to the set C_i containing j_k, and we simply identify C_i with the i-th element of $\operatorname{Cl}(\mathcal{O}')/C$ as enumerated by $\boldsymbol{\gamma}$ (in the lexicographic ordering of $\boldsymbol{\gamma}$-representations).

Thus we can compute the polynomials $\phi_i(X) = \Phi_\ell(X, y_i)$ as we enumerate $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ by accumulating a partial product of linear factors for each ϕ_i. But since our goal is to evaluate $z_i = \phi_i(\hat{\jmath}) \bmod p$, we simply substitute $x = \hat{\jmath} \bmod p$ into each linear factor, as we compute it, and accumulate the partial product in z_i.

Having computed the values z_i for $1 \le i \le \ell + 2$, we interpolate the unique polynomial $\phi(Y)$ of degree at most $\ell + 1$ for which $\phi(y_i) = z_i$, using Lagrange interpolation. This polynomial must be $\Phi_\ell(\hat{\jmath}, Y)$. We now give the algorithm.

Algorithm 2.1.

```
Input:   An odd prime $\ell$, a suitable order $\mathcal{O}$, a suitable prime $p$, and $x \in \mathbb{F}_p$.
Output:  The polynomial $\phi(Y) = \Phi_\ell(x, Y) \in \mathbb{F}_p[Y]$.

1. Compute presentations $\boldsymbol{\alpha}$ of $\mathrm{Cl}(\mathcal{O})$ and $\boldsymbol{\beta}$ of $\mathrm{Cl}(\mathcal{O}')$ suitable for $p$.
2. Represent generators of the subgroup $C \subset \mathrm{Cl}(\mathcal{O}')$ defined above in terms of $\boldsymbol{\beta}$.
3. Compute the presentation $\boldsymbol{\gamma}$ of $\mathrm{Cl}(\mathcal{O}')/C$ derived from $\boldsymbol{\beta}$, via Algorithm 3.
4. Find a root $w_1$ of $H_{\mathcal{O}} \bmod p$ (compute $H_{\mathcal{O}} \bmod p$ if needed).
5. Enumerate $\mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ as $w_1, w_2, \dots, w_{h(\mathcal{O})}$ using $\boldsymbol{\alpha}$.
6. Obtain $j_1 \in \mathrm{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ from $w_1$ using Vélu's algorithm.
7. Set $z_i \leftarrow 1$ and $y_i \leftarrow$ null for $1 \le i \le \ell + 2$.
8. For each $j_k = \delta_k j_1$ in $\mathrm{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ enumerated using $\boldsymbol{\beta}$:
   (a) Compute the index $i$ of $\delta_k$ in the $\boldsymbol{\gamma}$-enumeration of $\mathrm{Cl}(\mathcal{O}')/C$. If $i > \ell + 2$
       then proceed to the next $j_k$, skipping steps (b) and (c) below.
   (b) If $y_i =$ null then set $y_i$ to the $\ell$-parent of $j_k$ (via Vélu's algorithm) and
       for each $\ell$-sibling $\tilde{\jmath}$ of $y_i$ in $\mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ set $z_i \leftarrow z_i(x - \tilde{\jmath})$.
   (c) Set $z_i \leftarrow z_i(x - j_k)$.
9. Interpolate $\phi \in \mathbb{F}_p[Y]$ such that $\deg \phi \le \ell + 1$ and $\phi(y_i) = z_i$ for $1 \le i \le \ell + 2$.
10. Output $\phi$.
```

The value null assigned to y_i in step 7 is used to indicate that the value of y_i is not yet known. Each y_i is eventually set to a distinct $w_j \in \mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$.

Remark. In practical implementations, Algorithm 2 selects the primes $p \in S$ so that the presentations $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ are the same for every p and precomputes them (the only reason they might not be the same is the presence of prime ideals whose norm divides $v = v(p)$, but in practice we fix $v \le 2$, as discussed in Section 2.6).

Proposition 7. *Algorithm 2.1 outputs* $\phi(Y) = \Phi_\ell(x, Y) \bmod p$.

Proof. Let $\varphi(Y) = \Phi_\ell(x, Y)$. It follows from the discussion above that Algorithm 2.1 computes $z_i = \Phi_\ell(x, y_i)$ for $1 \le i \le \ell + 2$. Thus $\phi(y_i) = z_i = \varphi(y_i)$ for $\ell + 2$ values $y_i \in \mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$, and these values are necessarily distinct. The polynomials ϕ and φ both have degree at most $\ell + 1$, therefore $\phi = \varphi$. □

Theorem 8. *Assume the GRH. Algorithm 2.1 runs in $O(\ell^2 n^2 \log^2 n \operatorname{llog}^2 n)$ expected time using $O(\ell n)$ space, where $n = \log p$.*

Proof. The time complexity is dominated by step 8, which enumerates the $O(\ell^2)$ elements of $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ using $\boldsymbol{\beta}$. By [8, Theorem 5.1] and the suitability of $\mathcal{O}$ and p, we may assume each $\beta_i = [\mathfrak{b}_i]$, where $\mathfrak{b}_i$ has prime norm $b_i = O(\log n \operatorname{llog} n)$. Using Kronecker substitution and probabilistic root-finding [21], the expected time to find the (at most 2) roots of $\Phi_{b_i}(j_k, Y)$ is $O(n\mathsf{M}(n \log n \operatorname{llog} n))$, which dominates the cost for each j_k. Applying $\mathsf{M}(n) = O(n \log n \operatorname{llog} n)$ and multiplying by ℓ^2 yields the desired time bound. Taking into account $h(\mathcal{O}) = O(\ell)$ and $p > \ell$, the computation of $H_{\mathcal{O}} \bmod p$ uses $O(\ell n)$ space, by [42, Theorem 1], and this bounds the total space. □

3.4. ***A hybrid approach.*** Algorithm 2 achieves an essentially optimal space complexity, but its time complexity is attractive only when $\log q$ is not too large, say $\log q = O(\log^2 \ell)$. Algorithm 1 has an excellent time complexity, but achieves an optimal space complexity only when $\log q$ is very large, say $\log q = \Omega(\ell \log \ell)$. To address the intermediate range, we present a hybrid approach suggested by Daniel Kane that has the same space complexity as Algorithm 2 and a time complexity that is within a polylogarithmic factor of the time complexity of Algorithm 1.

The strategy is to replace the computation of $\bar{\phi}(Y) = \sum_{i,j} a_{ij}\hat{x}_i Y^j \bmod p$ in step 5 of Algorithm 1 with Algorithm 2.2 below. Algorithm 2.2 is similar to Algorithm 2.1, but rather than accumulating $\ell + 2$ values z_i in parallel, we compute them individually by enumerating the each of the sets C_i of children y_i in turn.

Algorithm 2.2.

```
Input:   An odd prime $\ell$, suitable order $\mathcal{O}$, suitable prime $p$, and $x_1, \dots, x_{\ell+1} \in \mathbb{F}_p$.
Output:  $\phi(Y) = \sum_{i,j} a_{ij} x_i Y^j \in \mathbb{F}_p[Y]$, where $\Phi_\ell(X,Y) = \sum_{i,j} a_{ij} X^i Y^j$.
1. Compute presentations $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ as in Algorithm 2.1.
2. Find a root $y_1$ of $H_{\mathcal{O}} \bmod p$ (compute $H_{\mathcal{O}} \bmod p$ if needed).
3. Enumerate $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_q)$ as $y_1, y_2, \dots, y_{h(\mathcal{O})}$ using $\boldsymbol{\alpha}$.
4. Obtain $j_1 \in \operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_q)$ from $y_1$ using Vélu's algorithm.
5. For $i$ from 1 to $\ell + 2$ do the following:
   (a) Use $\boldsymbol{\alpha}$ to compute the set $S_i$ of siblings of $y_i$ in $\operatorname{Ell}_{\mathcal{O}}(\mathbb{F}_p)$.
   (b) Use $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ to compute the set $C_i$ of children of $y_i$ in $\operatorname{Ell}_{\mathcal{O}'}(\mathbb{F}_p)$ (see below).
   (c) Compute $\phi_i(X) = \prod_{\tilde{j} \in S_i}(X - \tilde{j}) \prod_{\tilde{j} \in C_i}(X - \tilde{j}) = \sum c_{ik} X^k \bmod p$.
   (d) Compute $z_i = \sum_k c_{ik} x_k$.
6. Interpolate $\phi \in \mathbb{F}_p[Y]$ such that $\deg \phi \le \ell + 1$ and $\phi(y_i) = z_i$ for $1 \le i \le \ell + 2$.
7. Output $\phi$.
```

To compute the set C_i in step 5(b), for each $\tilde{j} \in C_i$ we determine the $\delta \in \mathrm{Cl}(\mathcal{O}')$ for which $\tilde{j} = \delta j_1$. Under the GRH, it follows from [11, Theorem 2.1] that we can express δ in the form $\delta = [\mathfrak{p}_1 \cdots \mathfrak{p}_t]$, where the ideals $\mathfrak{p}_i$ have prime norms bounded by $\log^c \ell$, for any $c > 2$, with $t = O(\log \ell)$. Assuming $\log p = O(\log \ell)$, this implies that we can compute each $\tilde{j}$ in $O(\log^{6+\epsilon} \ell)$ expected time, for any $\epsilon > 0$.

Proposition 9. *Algorithm 2.2 outputs $\phi(Y) = \sum_{i,j} a_{ij} x_i Y^j$.*

Proof. Let $\varphi(y) = \sum_{i,j} a_{ij} x_i Y^j$. The roots of $\phi_i(X)$ are the roots of $\Phi_\ell(X, y_i)$, thus $\sum_k c_{ik} X_k = \sum_{k,j} a_{kj} X^k y_i^j$, and we have $c_{ik} = \sum_j a_{kj} y_i^j$. It follows that $\phi(y_i) = z_i = \sum_k \sum_j a_{kj} x_k y_i^j = \varphi(y_i)$. Since $\phi(Y)$ and $\varphi(Y)$ both have degree at most $\ell + 1$ and agree at $\ell + 2$ distinct values y_i, they must be equal. □

Theorem 10. *Assume the GRH and that* $\log q = O(\ell \log \ell)$. *If Algorithm 1 uses Algorithm 2.2 to compute $\overline{\phi}(Y)$ in step 5, its expected running time is $O(\ell^3 \log^{6+\epsilon} \ell)$ using $O(\ell \log q + \ell \log \ell)$ space.*

Proof. It suffices to show that if $\log p = O(\log \ell)$, then Algorithm 2.2 runs in $O(\ell^2 \log^{6+\epsilon} \ell)$ expected time using $O(\ell \log \ell)$ space. The space bound is clear. For the time bound, the cost of step 5(b) is $O(\ell \log^{6+\epsilon} \ell)$ (see above), yielding an $O(\ell^2 \log^{6+\epsilon} \ell)$ bound on the expected time for step 5, which dominates. □

The extra logarithmic factors make the hybrid approach significantly slower than Algorithm 1 in practice, but it does allow us to achieve an essentially optimal space complexity with a quasicubic running time across the entire range of parameters.

3.5. *Computing a polycyclic presentation for a quotient group.*

We now give a generic algorithm to derive a polycyclic presentation $\boldsymbol{\gamma}$ for a quotient of finite abelian groups G/H. This presentation can be used to efficiently compute in G/H, and to compute the image of elements of G, as required by Algorithm 2.1.

Algorithm 3.

Input: A minimal polycyclic presentation $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)$ for a finite abelian group G and a subgroup $H = \langle \alpha_1, \ldots, \alpha_t \rangle$, with each α_i specified in terms of $\boldsymbol{\beta}$.

Output: A polycyclic presentation $\boldsymbol{\gamma}$ for G/H, with $\gamma_i = [\beta_i]$ for each $\beta_i \in \boldsymbol{\beta}$.

1. Derive a polycyclic presentation $\boldsymbol{\alpha}$ for H from $\alpha_1, \ldots, \alpha_t$ by using [42, Algorithm 2.2].
2. Enumerate H using $\boldsymbol{\alpha}$ and create a lookup table T_H to test membership in H.
3. Derive a polycyclic presentation $\boldsymbol{\gamma}$ for G/H from $[\beta_1], \ldots, [\beta_k]$ by using [42, Algorithm 2.2], using T_H as described below.
4. Output $\boldsymbol{\gamma}$, with relative orders $r(\boldsymbol{\gamma})$ and relations $s(\boldsymbol{\gamma})$.

The polycyclic presentation $\boldsymbol{\gamma}$ output by Algorithm 3 is not necessarily minimal. It can be converted to a minimal presentation by simply removing those γ_i with $r(\gamma_i) = 1$, but for the purpose of computing the image in G/H of elements of G represented in terms of $\boldsymbol{\beta}$, it is better not to do so.

Algorithm 2.2 of [42] requires a TABLELOOKUP function that searches for a given group element in a table of distinct group elements. In Algorithm 3 above, the elements of G are uniquely represented by their $\boldsymbol{\beta}$-representations, but elements of G/H are represented as equivalence classes $[\delta]$, with $\delta \in G$, which is not a unique representation. To implement the TABLELOOKUP function for G/H, we do the following: Given $[\delta_0] \in G/H$ and a table $T_{G/H}$ of distinct elements $[\delta_i]$ in G/H, we test whether $\delta_0\delta_i^{-1} \in H$, for each $[\delta_i] \in T$. With a suitable implementation of T_H (such as a hash table or balanced tree), membership in H can be tested in $O(\log|G|)$ time, which is dominated by the $O(\log^2|G|)$ time to compute $\delta_0\delta_i^{-1}$.

Once Algorithm 3 completes, the problem of uniquely representing elements of G/H is solved: Every element of G/H has a unique $\boldsymbol{\gamma}$-representation.

Theorem 11. *Algorithm 3 runs in $O(n \log^2 n)$ time using $O((m + n/m) \log n)$ space, where $n = |G|$ and $m = |H|$.*

Proof. The time complexity is dominated by the n/m calls to the TABLELOOKUP function performed by [42, Algorithm 2.2] in step 3, each of which performs m operations in G (using $\boldsymbol{\beta}$-representations) and m lookups in T_H, yielding a total cots of $O(n \log^2 n)$. The space bound is the size of T_H plus the size of $T_{G/H}$. □

3.6. ***Other modular functions.*** For a modular function g of level N and a prime $\ell \nmid N$, the modular polynomial Φ_ℓ^g is the minimal polynomial of the function $g(\ell z)$ over the field $\mathbb{C}(g)$. For suitable functions g, the isogeny volcano algorithm for computing $\Phi_\ell(X, Y)$ can be adapted to compute $\Phi_\ell^g(X, Y)$, as described in [8, §7]. There are some restrictions: Φ_ℓ^g must have degree $\ell + 1$ in both X and Y, and we require some additional constraints on the suitable orders $\mathcal{O}$ that we use. Specifically, we require that there is a generator τ of $\mathcal{O}$ for which $g(\tau)$ lies in the ring class field $K_{\mathcal{O}}$. In this case we say that $g(\tau)$ is a *class invariant*, and we let $H_{\mathcal{O}}^g(X)$ denote its minimal polynomial over K; see [7; 14; 16] for algorithms to compute $H_{\mathcal{O}}^g(X)$. We also require the polynomial $H_{\mathcal{O}}^g$ to be defined over $\mathbb{Z}$.

With this setup, there is then a one-to-one correspondence between the roots $j(\tau)$ of $H_{\mathcal{O}}$ and the roots $g(\tau)$ of $H_{\mathcal{O}}^g$ in which $\Psi^g(g(\tau), j(\tau)) = 0$, where Ψ^g is the minimal polynomial of g over $\mathbb{C}(j)$; note that Ψ^g does not depend on ℓ and is assumed to be given. The class group $\mathrm{Cl}(\mathcal{O}) \simeq \mathrm{Gal}(K_{\mathcal{O}}/K)$ acts compatibly on both sets of roots, and this allows us to compute Φ_ℓ^g modulo suitable primes p using essentially the same algorithm that is used to compute $\Phi_\ell \bmod p$. In particular, we can enumerate the set $\mathrm{Ell}_{\mathcal{O}}^g(\mathbb{F}_p) = \{x \in \mathbb{F}_p : H_{\mathcal{O}}^g(x) = 0\}$ using a polycyclic presentation $\boldsymbol{\alpha}$ for $\mathrm{Cl}(\mathcal{O})$, provided that we exclude from $\boldsymbol{\alpha}$ generators whose norm divides the level of g, and similarly for $\mathrm{Ell}^g_{\mathcal{O}'}(\mathbb{F}_p)$, where $\mathcal{O}' = \mathbb{Z} + \ell\mathcal{O}$.

Thus Algorithms 1 and 2 can both be adapted to compute instantiated modular polynomials $\phi^g(Y) = \Phi^g_\ell(x, Y) \bmod q$. Some effort may be required to determine the correspondence between $\mathrm{Ell}_{\mathcal{O}}(\mathbb{F}_p)$ and $\mathrm{Ell}^g_{\mathcal{O}}(\mathbb{F}_p)$ in cases where $\Psi^g(X, j(E))$ or $\Psi^g(g(E), Y)$ has multiple roots in $\mathbb{F}_p$; this issue arises when we need to compute a child or parent using Vélu's algorithm. There are several techniques for resolving such ambiguities; see [8, §7.3] and especially [16], which explores this issue in detail.

We emphasize that the point x at which we are evaluating $\Phi^g_\ell(x, Y)$ may be *any* element of $\mathbb{F}_q$; it need not correspond to the "g-invariant" of an elliptic curve.[4] This permits a very useful optimization that speeds up our original version of Algorithm 1 for computing $\phi_\ell(Y) = \phi^j_\ell(Y)$ by a factor of at least 9, as we now explain.

3.7. ***Accelerating the computation of $\boldsymbol{\phi_\ell(Y)}$ using $\boldsymbol{\gamma_2}$.*** Let $\gamma_2(z)$ be the unique cube root of $j(z)$ with integral Fourier expansion, a modular function of level 3 that yields class invariants for $\mathcal{O}$ whenever $3 \nmid \operatorname{disc}\mathcal{O}$. As noted in [8, §7.2], for $\ell > 3$ the modular polynomial $\Phi^{\gamma_2}_\ell$ can be written as

$$\Phi^{\gamma_2}_\ell(X, Y) = R(X^3, Y^3)Y^e + S(X^3, Y^3)XY + T(X^3, Y^3)X^2Y^{2-e}, \quad (5)$$

with $e = \ell + 1 \bmod 3$ and $R, S, T \in \mathbb{Z}[X, Y]$. We then have the identity

$$\Phi_\ell = R^3Y^e + (S^3 - 3RST)XY + TX^2Y^{2-e}. \quad (6)$$

When computing $\Phi^{\gamma_2}_\ell \bmod p$ with the isogeny volcano algorithm, one can exploit (5) to speed up the computation by at least a factor of 3. In addition, the integer coefficients of $\Phi^{\gamma_2}_\ell$ are also smaller than those of Φ_ℓ by roughly a factor of 3; we may use the height bound $h(\Phi^{\gamma_2}_\ell) \le 2\ell\log\ell + 8\ell$ from [8, Equation 18].

Let us consider how we may modify Algorithm 1 to exploit (6), thereby accelerating the computation of $\phi_\ell(Y) = \Phi_\ell(x, Y) \bmod q$, where $x = j(E) \in \mathbb{F}_q$. Let $r(Y) = R(x, Y) \bmod q$, and similarly define s and t in terms of S and T. Rather than computing $\Phi_\ell \bmod p$ in step 5(a), we compute $\Phi^{\gamma_2}_\ell \bmod p$ and derive R, S, and T from (5). We then compute polynomials $\bar{r}$, $\bar{s}$, and $\bar{t}$ mod p instead of $\bar{\phi}$ in step 5(b). Finally, we recover r, s, and $t \bmod q$ in step 6 via the explicit CRT and output

$$\phi = r^3Y^e + x(s^3 - 3rst)Y + x^2t^3Y^{2-e} \quad (7)$$

[4] Every $x \in \mathbb{F}_q$ is $j(E)$ for some $E/\mathbb{F}_q$, and when E is ordinary, $j(E)$ is the reduction of some $j(\tau) = j(\hat{E})$ with $\mathbb{Z}[\tau] = \mathcal{O} \simeq \mathrm{End}(E)$. But $g(\tau)$ might not be a class invariant for this $\mathcal{O}$.

in step 7. Adjusting the height bound B appropriately, this yields a speedup of nearly a factor of 9. Note that we are not assuming $x = j(E)$ has a cube root in $\mathbb{F}_q$, or that $\operatorname{End}(E) \simeq \mathcal{O}$ satisfies $3 \nmid \operatorname{disc} \mathcal{O}$; the identity (7) holds for all x.

We can similarly compute ϕ_X and ϕ_{XX}. To simplify the formulas, let us define $U = (S^3 - 3RST)$ and $u = U(x, Y) \bmod q$. Define

$$r'(Y) = \frac{\partial R}{\partial X}(x, Y) \quad \text{and} \quad r''(Y) = \frac{\partial^2 R}{\partial X^2}(x, Y),$$

and similarly for s, t, and u. Note that u, u', and u'' can all be easily expressed in terms of $r, r', r'', s, s', s'', t, t'$, and t''. We replace the computation of $\bar{\phi}_X$ and $\bar{\phi}_{XX}$ in step 5(c) with analogous computations of $\bar{r}', \bar{r}'', \bar{s}', \bar{s}'', \bar{t}'$, and $\bar{t}''$ mod p. We then obtain $r, r', r'', s, s', s'', t, t'$, and t'' via the explicit CRT mod q and apply

$$\begin{aligned} \phi_X &= 3r^2 r' Y^e + (u + xu')Y + (2xt^3 + 3x^2t^2t')Y^{2-e}, \\ \phi_{XX} &= (6rr'r' + 3r^2r'')Y^e \\ &\qquad + (2u' + u'')Y + (2t^3 + 12xt^2t' + 6x^2tt't' + 3x^2t^2t'')Y^{2-e}. \end{aligned}$$

3.8. ***Normalized isogenies.*** We now explain how Algorithms 1 and 2 may be used to compute normalized isogenies ψ, first using j-invariants, and then using g-invariants. Throughout this section $j = j(E) \in \mathbb{F}_q$ denotes the j-invariant of a given elliptic curve $E/\mathbb{F}_q$, defined by $y^2 = x^3 + Ax + B$, and $\phi(Y) = \Phi_\ell(j, Y)$. We use $\tilde{\jmath} = j(\widetilde{E})$ to denote a root of $\phi(Y)$ in $\mathbb{F}_q$. Our goal is to compute an equation for the image of $\psi : E \to \widetilde{E}$, and the kernel polynomial $h_\ell(X)$ for ψ.

3.8.1. *Algorithm 1.* When computing ϕ, we also compute the optional outputs ϕ_X and ϕ_{XX}, and then

$$\phi_Y(Y) = \frac{d}{dY}\phi(Y), \quad \phi_{YY}(Y) = \frac{d}{dY}\phi_Y(Y), \quad \text{and} \quad \phi_{XY} = \frac{d}{dY}\phi_X(Y).$$

We then compute the quantities $\Phi_*(j, \tilde{\jmath}) = \phi_*(\tilde{\jmath})$, for $* = X, Y, XX, XY, YY$, as defined in Section 2.1, and apply Elkies's algorithm [19, Algorithm 27] to compute $\widetilde{E}$ and $h_\ell(X)$.

3.8.2. *Algorithm 2.* Having computed ϕ and obtained $\tilde{\jmath}$, we run Algorithm 2 *again*, this time with the input $\tilde{\jmath}$, obtaining $\tilde{\phi}(Y) = \Phi_\ell(\tilde{\jmath}, Y)$, which we now regard as $\tilde{\phi}(X) = \Phi_\ell(X, \tilde{\jmath})$, by the symmetry of Φ_ℓ. We then compute

$$\Phi_X(j, \tilde{\jmath}) = \left(\frac{d}{dX}\tilde{\phi}\right)(j) \quad \text{and} \quad \Phi_Y(j, \tilde{\jmath}) = \left(\frac{d}{dY}\phi\right)(\tilde{\jmath}),$$

as well as the quantities

$$j' = \frac{18B}{A}j, \quad \tilde{\jmath}' = \frac{-\Phi_X(j, \tilde{\jmath})}{\ell\Phi_Y(j, \tilde{\jmath})}j', \quad \tilde{m} = \frac{\tilde{\jmath}'}{\tilde{\jmath}}, \quad \text{and} \quad \tilde{k} = \frac{\tilde{\jmath}'}{1728 - \tilde{\jmath}},$$

as in [19, Algorithm 27]. The normalized equation for $\widetilde{E}$ is then

$$y^2 = x^3 + \frac{\ell^4 \widetilde{m}\widetilde{k}}{48}x + \frac{\ell^6 \widetilde{m}^2 \widetilde{k}}{864},$$

and the $\mathsf{fastElkies}'$ algorithm in [5] may be used to compute $h_\ell(X)$.

3.8.3. *Handling g-invariants.* We assume that $g(E)$ is known to be a class invariant (see Section 3.9 below). Let $g = g(E)$, $\phi^g(Y) = \Phi_\ell^g(g, Y)$, and let $\widetilde{g} = g(\widetilde{E})$ denote a root of $\phi^g(Y)$ in $\mathbb{F}_q$. In the case of Algorithm 1 we compute

$$\Phi_X^g(g, \widetilde{g}) = \phi_X^g(\widetilde{g}) \quad \text{and} \quad \Phi_Y^g(g, \widetilde{g}) = \left(\frac{d}{dY}\phi^g\right)(\widetilde{g}),$$

and in the case of Algorithm 2 we make a second call with input $\widetilde{g}$ to obtain $\widetilde{\phi}^g(X) = \Phi_\ell^g(X, \widetilde{g})$ as above. We then compute

$$\Phi_X^g(g, \widetilde{g}) = \left(\frac{d}{dX}\widetilde{\phi}^g\right)(g) \quad \text{and} \quad \Phi_Y^g(g, \widetilde{g}) = \left(\frac{d}{dY}\phi^g\right)(\widetilde{g}).$$

We assume the modular equation $\Psi_\ell^g(G, J) = 0$ relating $g(z)$ to $j(z)$ can be solved for $j(z)$ (for the $g(z)$ considered in [8], we have $\deg_J \Psi^g(G, J) \le 2$), and let $F(G)$ satisfy $\Psi_\ell^g(F(J), J) = 0$ and $F' = dF/dG$.

To compute the normalized equation for $\widetilde{E}$, we proceed as in Section 3.8.2, except now

$$\widetilde{j}' = \frac{-\Phi_X^g(g, \widetilde{g})F'(\widetilde{g})}{\ell\Phi_Y^g(g, \widetilde{g})/F'(g)} j'.$$

The $\mathsf{fastElkies}'$ algorithm in [5] may then be used to compute h_ℓ, or, in the case of Algorithm 1, one may derive the trace of h_ℓ using $\Phi_{XX}^g(g, \widetilde{g})$, $\Phi_{XY}^g(g, \widetilde{g})$, and $\Phi_{YY}^g(g, \widetilde{g})$ as in Section 3.8.1, and compute h_ℓ as usual. We omit the details.

3.9. ***Verifying that $g(E)$ is a class invariant.*** Let $E/\mathbb{F}_q$ be an elliptic curve that is not supersingular (see [44] for fast tests), with $\operatorname{End}(E) \simeq \mathbb{O}$. As in Section 3.6, we call an element $g(E)$ of $\mathbb{F}_q$ a *class invariant* if

(1) $H_{\mathbb{O}}^g(X)$ splits into linear factors in the ring class field of O, and

(2) $g(E)$ is a common root of $H_{\mathbb{O}}^g(X)$ and $\Psi^g(X, j(E))$.

For practical applications, we would like to determine whether $g(E)$ is a class invariant without computing $\mathbb{O}$ (indeed, the application may be to compute $\mathbb{O}$). This is often easy to do, at least as far as condition (1) is concerned. As noted in Section 3.6, condition (1) can typically be guaranteed by constraints involving $D = \operatorname{disc}\mathbb{O}$ and the level N of g. Verifying condition (2) is more difficult, in general, but it can be easily addressed in particular cases if we know that $\Psi^g(X, j(E))$

either has a unique root in $\mathbb{F}_q$ (which then must also be a root of $H^g(\mathcal{O})$ once condition (1) is satisfied), or that all its roots in $\mathbb{F}_q$ are roots of $H^g(\mathcal{O})$, or of $H^{\bar{g}}(\mathcal{O})$ for some $\bar{g}$ with $\Phi_\ell^{\bar{g}} = \Phi_\ell^g$. In the latter case we may not determine $g(E)$ uniquely, but for the purposes of computing a normalized ℓ-isogeny this does not matter, any choice will work.

Taking $\gamma_2 = \sqrt[3]{j}$ as an example, condition (1) holds when $\left(\frac{D}{3}\right) \neq 0$, which means $j(E)$ is on the surface of its 3-volcano and has either 0 or 2 siblings. This can be easily determined using [18] or [42, §4.1]. If we have $q \equiv 2 \bmod 3$, the polynomial $\Psi^g(X, j(E)) = X^3 - j(E)$ has a unique root $g(E)$ in $\mathbb{F}_q$ and condition (2) also holds. (There are techniques to handle $q \equiv 1 \bmod 3$ — see [7], for example — but they assume that $\mathcal{O}$ is known.)

As a second example, consider the Weber $\mathfrak{f}$-function, which is related to the j-function by $\Psi^{\mathfrak{f}}(X, J) = (X^{24} - 16)^3 - X^{24}J$. Now we require $\left(\frac{D}{3}\right) \neq 0$ and $\left(\frac{D}{2}\right) = 1$. The latter is equivalent to $j(E)$ being on the surface of its 2-volcano with 2 siblings. If we also have $q \equiv 11 \bmod 12$, then $\Psi^{\mathfrak{f}}(X, j(E))$ has exactly two roots $\mathfrak{f}(E)$ and $-\mathfrak{f}(E)$, by [8, Lemma 7.3], and either may be used since $\Phi_\ell^{\mathfrak{f}} = \Phi_\ell^{-\mathfrak{f}}$.

For a more general solution, having verified condition (1), we may simply compute instantiated polynomials $\phi(Y) = \Phi_\ell(x, Y)$ for *every* root x of $\Psi^g(X, j(E))$ in $\mathbb{F}_q$. This can be done at essentially no additional cost, and we may then attempt to compute a normalized isogeny corresponding to each root x, which we validate by computing the dual isogeny (using the normalization factor $c = \ell$ rather than 1) and checking whether the composition corresponds to scalar multiplication by ℓ using randomly generated points in $E(\mathbb{F}_q)$. The cost of these validations is negligible compared to the cost of computing $\phi(Y)$ for even one x.

As a final remark, we note that in applications such as point counting where one is only concerned with the isogeny class of E, in cases where condition (1) is not satisfied, one may be able to obtain an isogenous $\widetilde{E}$ for which condition (1) holds by simply climbing to the surface of the relevant ℓ_0-volcanoes for the primes $\ell_0 \mid N$ (we regard N as fixed so ℓ_0 is small; $\ell_0 = 2, 3$ in the examples above).

4. Applications

In this section we analyze the use of Algorithms 1 and 2 in two particular applications: counting points and computing endomorphism rings.

Recall that for an elliptic curve $E/\mathbb{F}_q$, an odd prime ℓ is called an *Elkies prime* whenever $\phi(Y) = \Phi_\ell(j(E), Y)$ has a root in $\mathbb{F}_q$. This holds if and only if $t^2 - 4q$ is a square mod ℓ, where $t = q + 1 - \#E(\mathbb{F}_q)$. It follows from the Chebotarev density theorem that the set of Elkies primes for E has density $1/2$. The complexity of the Schoof-Elkies-Atkin algorithm [36] for computing $\#E(\mathbb{F}_q)$ depends critically on the number of *small* Elkies primes, specifically, the least $L = L(E)$ for which

$$\sum_{\text{Elkies primes } \ell \le L(E)} \log \ell > \log(4\sqrt{q}). \tag{8}$$

On average, one expects $L \approx \log q$, but even under the GRH the best proven bound is $L = O(\log^{2+\epsilon} q)$; see Appendix A of [34] by Satoh and Galbraith. This yields a complexity bound that is actually slightly *worse* than Schoof's original algorithm.

For practical purposes, the heuristic assumption $L(E) = O(\log q)$ is often used when analyzing the complexity of the SEA algorithm. This assumption holds for almost all elliptic curves [38], but it is known to fail in infinitely many cases [37]. We instead adopt the following weaker heuristic.

Heuristic 12. There exists a constant c such that for all sufficiently large q we have $L(E) \le c \log q \operatorname{llog} q$ for every elliptic curve $E/\mathbb{F}_q$.

Theorem 13. *Assume the GRH and Heuristic 12. Let $E/\mathbb{F}_q$ be an elliptic curve over a prime field $\mathbb{F}_q$ and let $n = \log q$. There is a Las Vegas algorithm to compute $\#E(\mathbb{F}_q)$ that runs in $O(n^4 \log^3 n \operatorname{llog} n)$ expected time using $O(n^2 \log n)$ space.*

Proof. Apply the SEA algorithm, using Algorithm 1 to compute $\phi(Y) = \Phi_\ell(j(E), Y)$ (and also ϕ_X and ϕ_{XX}), and ignore the Atkin primes, as in [38, Algorithm 1], for example. There are $O(n/\log n)$ primes in the sum (8), and under Heuristic 12, they are bounded by $L = O(n \log n)$. It follows from [38, Table 1] that the expected time to process each Elkies prime given ϕ is $O(n^3 \log^3 n \operatorname{llog}^2 n)$, which is dominated by the time to compute ϕ, as is the space. The theorem then follows from Theorem 4. □

A common application of the SEA algorithm is to search for random curves of prime (or near prime) order, for use in cryptographic applications. As shown in [38], we no longer need Heuristic 12 to do this; we can assume $L(E) = O(\log q)$ for a randomly chosen elliptic curve. Additionally, since we expect to count points on many curves ($\approx \log q$), we can take advantage of *batching*, whereby we extend Algorithm 1 to take multiple inputs $j(E_1) \in \mathbb{F}_{q_1}, \dots, j(E_k) \in \mathbb{F}_{q_k}$ and produce corresponding outputs for each (the $\mathbb{F}_{q_i}$ may coincide, but they need not). Provided $k = O(\log \ell)$, this does not change the time complexity (relative to the largest $\mathbb{F}_{q_i}$), since the most time-consuming steps depend only on ℓ, not $j(E)$, and the space complexity is increased by at most a factor of k.[5]

Let $E_{a,b}$ denote the elliptic curve defined by $y^2 = x^3 + ax + b$, and for any real number $x > 3$, let $T(x)$ denote the set of all triples (q, a, b) with $q \in [x, 2x]$ prime, $a, b \in \mathbb{F}_q$, and $\#E_{a,b}$ prime. The following result strengthens [38, Theorem 3].

[5] These remarks also apply to Algorithm 2.

Theorem 14. *There is a Las Vegas algorithm that, given x, outputs a random triple $(q,a,b) \in T(x)$ and the prime $\#E_{a,b}(\mathbb{F}_q)$, with q uniformly distributed over the primes in $[x,2x]$ and (a,b) uniformly distributed over the pairs $(c,d) \in \mathbb{F}_q^2$ for which $\#E_{c,d}(\mathbb{F}_q)$ is prime. Under the GRH, its expected running time is $O(n^5 \log^2 n \operatorname{llog} n)$ using $O(n^2 \log^2 n)$ space, where $n = \log x$.*

Proof. We modify the algorithm in [38] to use Algorithm 1, operating on batches of $O(\log n)$ inputs at a time. One then obtains an $O(n^4 \log n \operatorname{llog} n)$ bound on the average time to compute $\#E_{a,b}(\mathbb{F}_q)$ for primes $q \in [x,2x]$, and a space complexity of $O(n^2 \log^2 n)$. The theorem then follows from the proof of [38, Theorem 3]. □

A second application of Algorithms 1 and 2 is in the computation of the endomorphism ring of an ordinary elliptic curve. The algorithm in [3] achieves a heuristically subexponential running time of $L[1/2, \sqrt{3}/2]$ using $L[1/2, 1/\sqrt{3}]$ space. Algorithms 1 and 2 both improve the space complexity bound to $L[1/2, 1/\sqrt{12}]$, which is significant, since space is the limiting factor in these computations. Algorithm 2 also provides a slight improvement to the time complexity that is not visible in the $L[\alpha, c]$ notation but may be practically useful. These remarks also apply to the algorithm in [27] for evaluating isogenies of large degree.

5. Computations

Using a modified version of the SEA algorithm incorporating Algorithm 1, we determined the number of points on the elliptic curve

$$y^2 = x^3 + 2718281828x + 3141592653,$$

modulo the 5011-digit prime $q = 16219299585 \cdot 2^{16612} - 1$. The algorithm ignored the Atkin primes and computed the trace of Frobenius t modulo 700 Elkies primes, the largest of which was $\ell = 11681$; see [41] for details, including the exact value of t, which is too large to print here. The computation was distributed over 32 cores (3.0 GHz AMD Phenom II), and took about 6 weeks. Table 1 gives the time taken for various parts of the computation.

For $\ell = 11681$, the size of $\phi_\ell^{\mathfrak{f}}(Y) = \Phi_\ell^{\mathfrak{f}}(\mathfrak{f}(E), Y)$ was under 20MB and took about two hours to compute on a single core. As can be seen in Table 1, the computation of $\phi_\ell^{\mathfrak{f}}$ accounted for less than 3% of the total running time, despite being the asymptotically dominant step. This is primarily due to the use of the Weber $\mathfrak{f}$-invariant; with a less advantageous invariant (in the worst case, the j-invariant with the optimization of Section 3.7), the time spent computing ϕ_ℓ would have been comparable to or greater than the time spent on the remaining steps. But in any case the computation would still have been quite feasible.

To demonstrate the scalability of the algorithm, we computed $\phi_\ell^{\mathfrak{f}}(Y)$ for an elliptic curve $E/\mathbb{F}_q$, with $\ell = 100019$ and $q = 2^{86243} - 1$. Running on 32 cores

Task	CPU days
Computing $\phi_\ell^{\mathfrak{f}}(Y)$ with Algorithm 1	32
Computing $X^q \bmod \phi_\ell$ (using [24])	995
Computing h_ℓ using [19, Algorithm 27]	3
Computing Y^q and X^q mod h_ℓ, E using [22]	326
Computing the eigenvalue λ_ℓ using BSGS	22
Total	1378

Table 1. Breakdown of time spent computing $\#E(\mathbb{F}_q)$ for a 5011-bit prime q. The computation was performed on 32 cores of a 3.0 GHz AMD Phenom II.

(Algorithms 1 and 2 are both easily parallelized), this computation took less than a week. We note that the size of the instantiated modular polynomial $\phi_\ell^{\mathfrak{f}}$ (and ϕ_ℓ) is almost exactly one gigabyte, whereas the size of $\Phi_\ell^{\mathfrak{f}}$ is many terabytes, and we estimate that the size of Φ_ℓ is around 20 or 30 petabytes.

Acknowledgments

I am grateful to David Harvey for his assistance with the algorithms for fast polynomial arithmetic used in the computations described in Section 5, and to Daniel Kane for suggesting the hybrid approach outlined in Section 3.4. I also thank the referees for their comments and helpful suggestions.

This work was partially supported by NSF grant DMS-1115455.

References

[1] Juliana Belding, Reinier Bröker, Andreas Enge, and Kristin Lauter, *Computing Hilbert class polynomials*, in van der Poorten and Stein [33], 2008, pp. 282–295. MR 2009j:11200

[2] Daniel J. Bernstein and Jonathan P. Sorenson, *Modular exponentiation via the explicit Chinese remainder theorem*, Math. Comp. **76** (2007), no. 257, 443–454. MR 2007f:11142

[3] Gaetan Bisson and Andrew V. Sutherland, *Computing the endomorphism ring of an ordinary elliptic curve over a finite field*, J. Number Theory **131** (2011), no. 5, 815–831. MR 2012a:11080

[4] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system, I: The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265. MR 1484478

[5] A. Bostan, F. Morain, B. Salvy, and É. Schost, *Fast algorithms for computing isogenies between elliptic curves*, Math. Comp. **77** (2008), no. 263, 1755–1778. MR 2009k:11207

[6] Reinier Bröker, *A p-adic algorithm to compute the Hilbert class polynomial*, Math. Comp. **77** (2008), no. 264, 2417–2435. MR 2009j:11093

[7] ———, *p-adic class invariants*, LMS J. Comput. Math. **14** (2011), 108–126. MR 2801172

[8] Reinier Bröker, Kristin Lauter, and Andrew V. Sutherland, *Modular polynomials via isogeny volcanoes*, Math. Comp. **81** (2012), no. 278, 1201–1231. MR 2012m:11180

[9] Reinier Bröker and Andrew V. Sutherland, *An explicit height bound for the classical modular polynomial*, Ramanujan J. **22** (2010), no. 3, 293–313. MR 2011g:11123

[10] D. A. Buell and J. T. Teitelbaum (eds.), *Computational perspectives on number theory: Proceedings of the conference in honor of A. O. L. Atkin held at the University of Illinois, Chicago, IL, September* 1995, AMS/IP Studies in Advanced Mathematics, no. 7, Providence, RI, American Mathematical Society, 1998. MR 98g:11001

[11] Andrew M. Childs, David Jao, and Vladimir Soukharev, *Constructing elliptic curve isogenies in quantum subexponential time*, 2012. arXiv 1012.4019v2 [quant-ph]

[12] Jean-Guillaume Dumas (ed.), *ISSAC* 2006: *Proceedings of the* 2006 *International Symbolic and Algebraic Computation held in Genova, July* 9–12, 2006, New York, ACM Press, 2006. MR 2289094

[13] Noam D. Elkies, *Elliptic and modular curves over finite fields and related computational issues*, in Buell and Teitelbaum [10], 1998, pp. 21–76. MR 99a:11078

[14] Andreas Enge, *The complexity of class polynomial computation via floating point approximations*, Math. Comp. **78** (2009), no. 266, 1089–1107. MR 2010h:11097

[15] ———, *Computing modular polynomials in quasi-linear time*, Math. Comp. **78** (2009), no. 267, 1809–1824. MR 2010b:11171

[16] Andreas Enge and Andrew V. Sutherland, *Class invariants by the CRT method*, in Hanrot et al. [23], 2010, pp. 142–156. MR 2012d:11246

[17] Claus Fieker and David R. Kohel (eds.), *Algorithmic number theory: Proceedings of the* 5*th International Symposium* (*ANTS-V*) *held at the University of Sydney, July* 7–12, 2002, Lecture Notes in Computer Science, no. 2369, Berlin, Springer, 2002. MR 2004j:11002

[18] Mireille Fouquet and François Morain, *Isogeny volcanoes and the SEA algorithm*, in Fieker and Kohel [17], 2002, pp. 276–291. MR 2005c:11077

[19] Steven D. Galbraith, *Mathematics of public key cryptography*, Cambridge University Press, 2012. MR 2931758

[20] Steven D. Galbraith, Florian Hess, and Nigel P. Smart, *Extending the GHS Weil descent attack*, in Knudsen [29], 2002, pp. 29–44. MR 2004f:94060

[21] Joachim von zur Gathen and Jürgen Gerhard, *Modern computer algebra*, 2nd ed., Cambridge University Press, 2003. MR 2004g:68202

[22] P. Gaudry and F. Morain, *Fast algorithm for computing the eigenvalue in the Schoof-Elkies-Atkin algorithm*, in Dumas [12], 2006, pp. 109–115. MR 2289108

[23] Guillaume Hanrot, François Morain, and Emmanuel Thomé (eds.), *Algorithmic number theory: Proceedings of the* 9*th Biennial International Symposium* (*ANTS-IX*) *held in Nancy, July* 19–23, 2010, Lecture Notes in Computer Science, no. 6197, Berlin, Springer, 2010. MR 2011g:11002

[24] David Harvey, *A cache-friendly truncated FFT*, Theoret. Comput. Sci. **410** (2009), no. 27–29, 2649–2658. MR 2010g:68327

[25] IEEE (ed.), *Proceedings of the* 49*th IEEE Symposium on Foundations of Computer Science held in Philadelphia, October* 25–28, 2008, Los Alamitos, CA, Institute of Electrical and Electronics Engineers, IEEE Computer Society, 2008.

[26] INRIA Project-Team TANC, 2009 *activity report*, 2009. http://raweb.inria.fr/rapportsactivite/RA2009/tanc/tanc.pdf

[27] David Jao and Vladimir Soukharev, *A subexponential algorithm for evaluating large degree isogenies*, in Hanrot et al. [23], 2010, pp. 219–233. MR 2011h:11144

[28] Kiran S. Kedlaya and Christopher Umans, *Fast Modular Composition in any Characteristic*, in IEEE [25], 2008, pp. 146–155.

[29] Lars Knudsen (ed.), *Advances in cryptology—EUROCRYPT 2002: Proceedings of the 21st International Annual Conference on the Theory and Applications of Cryptographic Techniques held in Amsterdam, April 28–May 2, 2002*, Lecture Notes in Computer Science, no. 2332, Berlin, Springer, 2002. MR 2003m:94074

[30] David Russell Kohel, *Endomorphism rings of elliptic curves over finite fields*, Ph.D. thesis, University of California, Berkeley, 1996, p. 117. http://search.proquest.com/docview/304241260 MR 2695524

[31] Serge Lang, *Elliptic functions*, 2nd ed., Graduate Texts in Mathematics, no. 112, Springer, New York, 1987. MR 88c:11028

[32] The PARI Group, *PARI/GP (version 2.4.3)*, 2011. http://pari.math.u-bordeaux.fr/

[33] Alfred J. van der Poorten and Andreas Stein (eds.), *Algorithmic number theory: Proceedings of the 8th International Symposium (ANTS-VIII) held in Banff, AB, May 17–22, 2008*, Lecture Notes in Computer Science, no. 5011, Berlin, Springer, 2008. MR 2009h:11002

[34] Takakazu Satoh, *On p-adic point counting algorithms for elliptic curves over finite fields*, in Fieker and Kohel [17], 2002, pp. 43–66. MR 2004k:11098

[35] A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing (Arch. Elektron. Rechnen) **7** (1971), 281–292. MR 45 #1431

[36] René Schoof, *Counting points on elliptic curves over finite fields*, J. Théor. Nombres Bordeaux **7** (1995), no. 1, 219–254. MR 97i:11070

[37] Igor Shparlinski, *On the product of small Elkies primes*, 2013. arXiv 1301.0035 [math.NT]

[38] Igor E. Shparlinski and Andrew V. Sutherland, *On the distribution of Atkin and Elkies primes*, 2011. arXiv 1112.3390 [math.NT]

[39] Joseph H. Silverman, *The arithmetic of elliptic curves*, Graduate Texts in Mathematics, no. 106, Springer, New York, 1986. MR 87g:11070

[40] ———, *Advanced topics in the arithmetic of elliptic curves*, Graduate Texts in Mathematics, no. 151, Springer, New York, 1999. MR 96b:11074

[41] Andrew V. Sutherland, *Genus 1 point counting records over prime fields*, 2010. http://math.mit.edu/~drew/SEArecords.html

[42] ———, *Computing Hilbert class polynomials with the Chinese remainder theorem*, Math. Comp. **80** (2011), no. 273, 501–538. MR 2011k:11177

[43] ———, *Accelerating the CM method*, LMS J. Comput. Math. **15** (2012), 172–204. MR 2970725

[44] ———, *Identifying supersingular elliptic curves*, LMS J. Comput. Math. **15** (2012), 317–325. MR 2988819

[45] Jacques Vélu, *Isogénies entre courbes elliptiques*, C. R. Acad. Sci. Paris Sér. A-B **273** (1971), A238–A241. http://gallica.bnf.fr/ark:/12148/bpt6k56191248/f52.image MR 45 #3414

ANDREW V. SUTHERLAND: drew@math.mit.edu
Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139, United States

VOLUME EDITORS

Everett W. Howe
Center for Communications Research
4320 Westerra Court
San Diego, CA 92121-1969
United States

Kiran S. Kedlaya
Department of Mathematics
University of California, San Diego
9500 Gilman Drive #0112
La Jolla, CA 92093-0112

Front cover artwork based on a detail of
Chicano Legacy 40 Años ©2010 Mario Torero.

Electronic copies can be obtained free of charge from http://msp.org/obs/1 and printed copies can be ordered from MSP (contact@msp.org).

The Open Book Series is a trademark of Mathematical Sciences Publishers.

ISSN: 2329-9061 (print), 2329-907X (electronic)

ISBN: 978-1-935107-00-2 (print), 978-1-935107-01-9 (electronic)

First published 2013.

msp

MATHEMATICAL SCIENCES PUBLISHERS
798 Evans Hall #3840, c/o University of California, Berkeley CA 94720-3840
contact@msp.org http://msp.org

THE OPEN BOOK SERIES 1

Tenth Algorithmic Number Theory Symposium

The Algorithmic Number Theory Symposium (ANTS), held biennially since 1994, is the premier international forum for research in computational number theory. ANTS is devoted to algorithmic aspects of number theory, including elementary, algebraic, and analytic number theory, the geometry of numbers, arithmetic algebraic geometry, the theory of finite fields, and cryptography.

This volume is the proceedings of the tenth ANTS meeting, held July 9–13, 2012, at the University of California, San Diego. It includes revised and edited versions of the 25 refereed papers presented at the conference, together with extended abstracts of two of the five invited talks.

TABLE OF CONTENTS