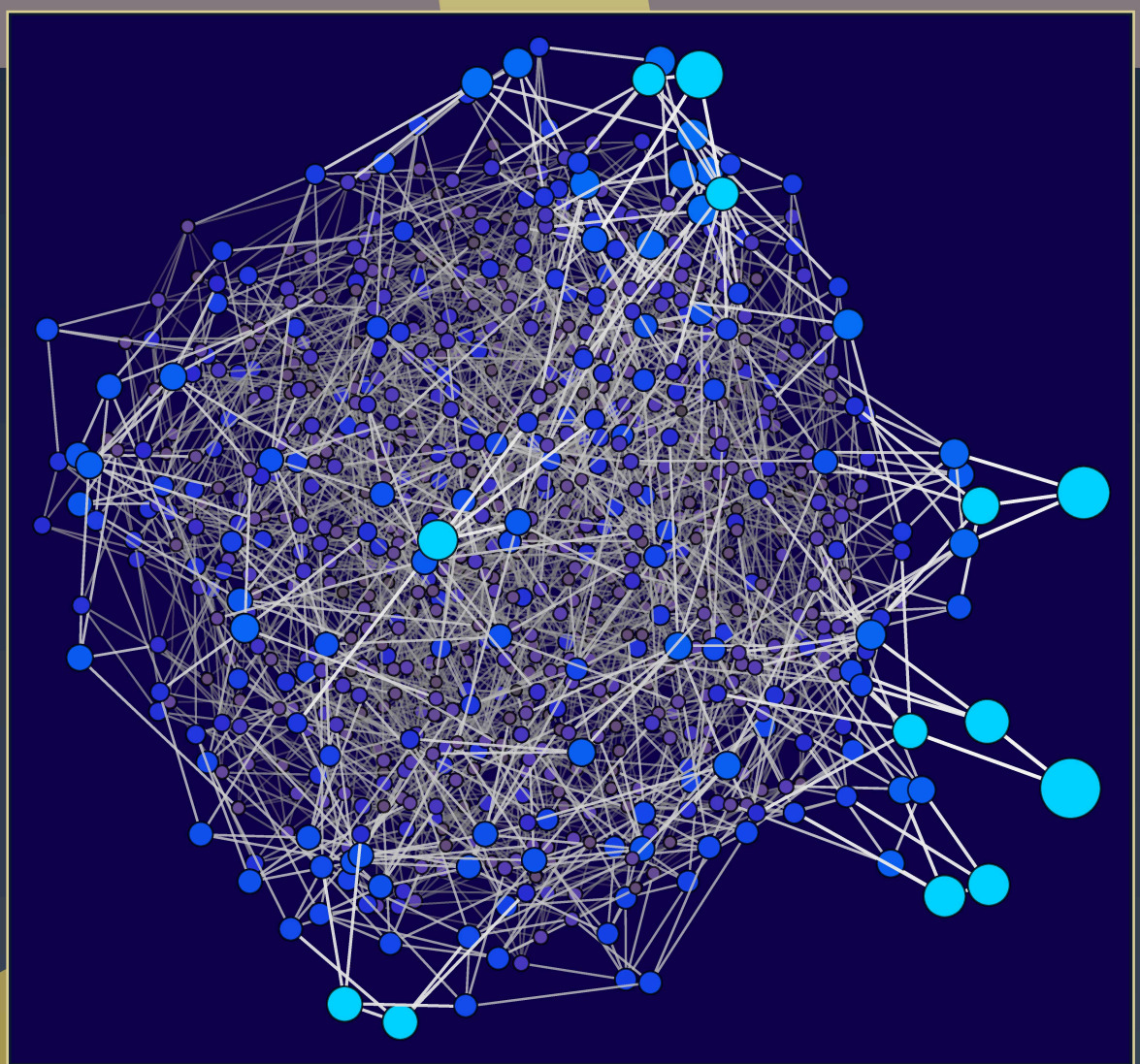
THE OPEN BOOK SERIES 4

ANTS XIV
Proceedings of the Fourteenth Algorithmic Number Theory Symposium

Faster computation of isogenies of large prime degree

Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith



msp

THE OPEN BOOK SERIES **4** (2020)
Fourteenth Algorithmic Number Theory Symposium
https://doi.org/10.2140/obs.2020.4.39

Faster computation of isogenies of large prime degree

Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith

Dedicated to the memory of Peter Lawrence Montgomery

Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve, and P a point in $\mathcal{E}(\mathbb{F}_q)$ of prime order ℓ. Vélu's formulæ let us compute a quotient curve $\mathcal{E}' = \mathcal{E}/\langle P\rangle$ and rational maps defining a quotient isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ in $\widetilde{O}(\ell)$ $\mathbb{F}_q$-operations, where the $\widetilde{O}$ is uniform in q. This article shows how to compute $\mathcal{E}'$, and $\phi(Q)$ for Q in $\mathcal{E}(\mathbb{F}_q)$, using only $\widetilde{O}(\sqrt{\ell})$ $\mathbb{F}_q$-operations, where the $\widetilde{O}$ is again uniform in q. As an application, this article speeds up some computations used in the isogeny-based cryptosystems CSIDH and CSURF.

1. Introduction

Let $\mathcal{E}$ be an elliptic curve over a finite field $\mathbb{F}_q$ of odd characteristic, and let P be a point in $\mathcal{E}(\mathbb{F}_q)$ of order n. The point P generates a cyclic subgroup $\mathcal{G} \subseteq \mathcal{E}(\mathbb{F}_q)$, and there exists an elliptic curve $\mathcal{E}'$ over $\mathbb{F}_q$ and a separable degree-n quotient isogeny

$$\phi : \mathcal{E} \longrightarrow \mathcal{E}' \qquad \text{with} \qquad \ker\phi = \mathcal{G} = \langle P\rangle\,;$$

the isogeny ϕ is also defined over $\mathbb{F}_q$. We want to compute $\phi(Q)$ for a point Q in $\mathcal{E}(\mathbb{F}_q)$ as efficiently as possible.

If n is composite, then we can decompose ϕ into a series of isogenies of prime degree. Computationally, this assumes that we can factor n, but finding a prime factor ℓ of n is not a bottleneck compared to the computation of an ℓ-isogeny by the techniques considered here. We thus reduce to the case where $n = \ell$ is prime.

Date: 2020.09.30.

For the long version of this article see [6]. Thanks to the anonymous referees for their comments. Author list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf. Part of this work was carried out while the first author was visiting the Simons Institute for the Theory of Computing. This work was supported by the Cisco University Research Program, by DFG Cluster of Excellence 2092 "CASA: Cyber Security in the Age of Large-Scale Adversaries", and by the U.S. National Science Foundation under grant 1913167. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation" (or other funding agencies). Permanent ID of this document: `44d5ade1c1778d86a5b035ad20f880c08031a1dc`.
MSC2020: 11Y16.
Keywords: isogenies, resultants.

Vélu introduced formulæ for ϕ and $\mathcal{E}'$ (see [56] and [38, §2.4]): for $\mathcal{E}$ defined by $y^2 = x^3 + a_2x^2 + a_4x + a_6$ and $\ell \geq 3$, we have

$$\phi : (X, Y) \longmapsto \left(\frac{\Phi_{\mathcal{G}}(X)}{\Psi_{\mathcal{G}}(X)^2}, \frac{Y\Omega_{\mathcal{G}}(X)}{\Psi_{\mathcal{G}}(X)^3} \right)$$

where

$$\begin{aligned}
\Psi_{\mathcal{G}}(X) &= \textstyle\prod_{s=1}^{(\ell-1)/2} \left(X - x([s]P)\right), \\
\Phi_{\mathcal{G}}(X) &= 4(X^3 + a_2X^2 + a_4X + a_6)(\Psi'_{\mathcal{G}}(X)^2 - \Psi''_{\mathcal{G}}(X)\Psi_{\mathcal{G}}(X)) \\
&\qquad - 2(3X^2 + 2a_2X + a_4)\Psi'_{\mathcal{G}}(X)\Psi(X) + (\ell X - \textstyle\sum_{s=1}^{\ell-1} x([s]P))\Psi_{\mathcal{G}}(X)^2, \\
\Omega_{\mathcal{G}}(X) &= \Phi'_{\mathcal{G}}(X)\Psi_{\mathcal{G}}(X) - 2\Phi_{\mathcal{G}}(X)\Psi'_{\mathcal{G}}(X).
\end{aligned}$$

The obvious way to compute $\phi(Q)$ is to compute the rational functions shown above, i.e., to compute the coefficients of the polynomials Ψ_G, Φ_G, Ω_G; and then evaluate those polynomials. This takes $\widetilde{O}(\ell)$ operations. (If we need the defining equation of $\mathcal{E}'$, then we can obtain it by evaluating $\phi(Q)$ for a few Q outside $\mathcal{G}$, possibly after extending $\mathbb{F}_q$, and then interpolating a curve equation through the resulting points. Alternatively, Vélu gives further formulas for the defining equation.) We emphasize, however, that the goal is not to compute the coefficients of these functions; the goal is to evaluate the functions at a specified point.

The core algorithmic problem falls naturally into a more general framework: the efficient evaluation of polynomials and rational functions over $\mathbb{F}_q$ whose roots are values of a function from a cyclic group to $\mathbb{F}_q$.

Fix a cyclic group $\mathcal{G}$ (which we will write additively), a generator P of $\mathcal{G}$, and a function $f : \mathcal{G} \to \mathbb{F}_q$. For each finite subset S of $\mathbb{Z}$, we define a polynomial

$$h_S(X) = \prod_{s \in S} (X - f([s]P)),$$

where $[s]P$ denotes the sum of s copies of P. The kernel polynomial $\Psi_{\mathcal{G}}(x)$ above is an example of this, with $f = x$ and $S = \{1, \ldots, (\ell - 1)/2\}$. Another example is the cyclotomic polynomial Φ_n, where f embeds $\mathbb{Z}/n\mathbb{Z}$ in the roots of unity of $\mathbb{F}_q$, and $\Phi_n(X) = h_S(X)$ where $S = \{i \mid 0 \leq i < n, \gcd(i, n) = 1\}$. More generally, if f maps $i \mapsto \zeta^i$ for some ζ, then $h_S(X)$ is a polynomial whose roots are various powers of ζ; similarly, if f maps $i \mapsto i\beta$ for some β, then $h_S(X)$ is a polynomial whose roots are various integer multiples of β.

Given f and S, then, we want to compute $h_S(\alpha) = \prod_{s \in S}(\alpha - f([s]P))$ for any α in $\mathbb{F}_q$. One can always directly compute $h_S(\alpha)$ in $O(\#S)$ $\mathbb{F}_q$-operations; this is the standard way to compute $\Psi_G(\alpha)$. But if S has enough additive structure, and if f is sufficiently compatible with the group structure on $\mathcal{G}$, then we can compute $h_S(\alpha)$ in $\widetilde{O}(\sqrt{\#S})$ $\mathbb{F}_q$-operations, as we will see in §2, §3, and §4. Our main theoretical result is Theorem 4.11, which shows how to achieve this quasi-square-root complexity for a large class of

S when f is the x-coordinate on an elliptic curve. We apply this to the special case of efficient ℓ-isogeny computation in §5. We discuss applications in isogeny-based cryptography in §6.

Most of this paper focuses on asymptotic exponents, in particular improving ℓ-isogeny evaluation from cost $\widetilde{O}(\ell)$ to cost $\widetilde{O}(\sqrt{\ell})$. However, this analysis hides polylogarithmic factors that can swamp the exponent improvement for small ℓ. In the full version [6], we instead analyze costs for concrete values of ℓ, and ask how large ℓ needs to be for the $\widetilde{O}(\sqrt{\ell})$ algorithms to outperform conventional algorithms.

1.1. ***Model of computation.*** We state our framework for $\mathbb{F}_q$ for concreteness. All time complexities are in $\mathbb{F}_q$-operations, with the O and $\widetilde{O}$ uniform over q.

The ideas are more general. The algorithms here are algebraic algorithms in the sense of [16], and can further be lifted to algorithms defined over $\mathbb{Z}[1/2]$ and in some cases over $\mathbb{Z}$. In other words, the algorithms are agnostic to the choice of q in $\mathbb{F}_q$, except for sometimes requiring q to be odd; and the algorithms can also be applied to more general rings, as long as all necessary divisions can be carried out.

Restricting to algebraic algorithms can damage performance. For example, for most input sizes, the fastest known algorithms to multiply polynomials over $\mathbb{F}_q$ are faster than the fastest known algebraic algorithms for the same task. This speedup is only polylogarithmic and hence is not visible at the level of detail of our analysis (the full version [6] contains a detailed analysis of concrete performances), but implementors should be aware that simply performing a sequence of separate $\mathbb{F}_q$-operations is not always the best approach.

2. Strassen's deterministic factorization algorithm

As a warmup, we review a deterministic algorithm that provably factors n into primes in time $\widetilde{O}(n^{1/4})$. There are several such algorithms in the literature using fast polynomial arithmetic, including [53], [12], [23], and [34]; there is also a separate series of lattice-based algorithms surveyed in, e.g., [4]. Strassen's algorithm from [53] has the virtue of being particularly simple, and is essentially the algorithm presented in this section.

The state of the art in integer factorization has advanced far beyond $\widetilde{O}(n^{1/4})$. For example, ECM [39], Lenstra's elliptic-curve method of factorization, is plausibly conjectured to take time $n^{o(1)}$. We present Strassen's algorithm because Strassen's main subroutine is the simplest example of a much broader speedup that we use.

2.1. ***Factorization via modular factorials.*** Computing $\gcd(n, \ell! \bmod n)$ reveals whether n has a prime factor $\leq \ell$. Binary search through all $\ell \leq \sqrt{n}$ then finds the smallest prime factor of n. Repeating this process completely factors n into primes.

The rest of this section focuses on the problem of computing $\ell! \bmod n$, given positive integers ℓ and n. The algorithm of §2.3 uses $\widetilde{O}(\sqrt{\ell})$ additions, subtractions, and multiplications in $\mathbb{Z}/n\mathbb{Z}$, plus negligible overhead. For comparison, a straightforward computation would use $\ell - 1$ multiplications modulo n. The $\widetilde{O}$ here is uniform over n.

2.2. ***Modular factorials as an example of the main problem.*** Define $\mathcal{G}$ as the additive group $\mathbb{Z}$, define $P = 1$, define $f : \mathcal{G} \to \mathbb{Z}/n\mathbb{Z}$ as $s \mapsto s$, and define $h_S(X) = \prod_{s \in S}(X - f([s]P)) \in (\mathbb{Z}/n\mathbb{Z})[X]$. Then, in particular, $h_S(X) = (X-1)\cdots(X-\ell)$ for $S = \{1, \ldots, \ell\}$, and one can compute $\ell! \bmod n$ by computing $h_S(\ell+1)$ or, alternatively, by computing $(-1)^\ell h_S(0)$. This fits the modular-factorials problem, in the special case that n is a prime number q, into the framework of §1.

2.3. ***An algorithm for modular factorials.*** Compute $b = \lfloor\sqrt{\ell}\rfloor$, and define $I = \{0, 1, 2, \ldots, b-1\}$. Use a product tree to compute the polynomial $h_I(X) = X(X-1)(X-2)\cdots(X-(b-1)) \in (\mathbb{Z}/n\mathbb{Z})[X]$.

Define $J = \{b, 2b, 3b, \ldots, b^2\}$. Compute $h_J(X)$, and then compute the resultant of $h_J(X)$ and $h_I(X)$. This resultant is $h_I(b)h_I(2b)h_I(3b)\cdots h_I(b^2)$, i.e., $(b^2)! \bmod n$.

One can compute the resultant of two polynomials via continued fractions; see, e.g., [54]. An alternative here, since h_J is given as a product of linear polynomials, is to use a remainder tree to compute $h_I(b), h_I(2b), \ldots, h_I(b^2) \in \mathbb{Z}/n\mathbb{Z}$, and then multiply. Either approach uses $\widetilde{O}(\sqrt{\ell})$ operations.

Finally, multiply by $(b^2+1)(b^2+2)\cdots\ell$ modulo n, obtaining $\ell! \bmod n$.

3. Evaluation of polynomials whose roots are powers

Pollard [49] introduced a deterministic algorithm that provably factors n into primes in time $O(n^{1/4+\epsilon})$. Strassen's algorithm from [53] was a streamlined version of Pollard's algorithm, replacing $O(n^{1/4+\epsilon})$ with $\widetilde{O}(n^{1/4})$.

This section reviews Pollard's main subroutine, a fast method to evaluate a polynomial whose roots (with multiplicity) form a geometric progression. For comparison, Strassen's main subroutine is a fast method to evaluate a polynomial whose roots form an arithmetic progression. See §2.3 above.

3.1. ***A multiplicative version of modular factorials.*** Fix $\zeta \in (\mathbb{Z}/n\mathbb{Z})^*$. Define $\mathcal{G} = \mathbb{Z}$, define $P = 1$, define $f : \mathcal{G} \to (\mathbb{Z}/n\mathbb{Z})^*$ as $s \mapsto \zeta^s$, and define $h_S(X) = \prod_{s \in S}(X - f([s]P)) = \prod_{s \in S}(X - \zeta^s) \in (\mathbb{Z}/n\mathbb{Z})[X]$. (For comparison, in §2, f was $s \mapsto s$, and $h_S(X)$ was $\prod_{s \in S}(X - s)$.)

In particular, $h_S(X) = \prod_{s=1}^{\ell}(X - \zeta^s)$ for $S = \{1, 2, 3, \ldots, \ell\}$. Given $\alpha \in \mathbb{Z}/n\mathbb{Z}$, one can straightforwardly evaluate $h_S(\alpha)$ for this S using $O(\ell)$ algebraic operations in $\mathbb{Z}/n\mathbb{Z}$. The method in §3.2 accomplishes the same result using only $\widetilde{O}(\sqrt{\ell})$ operations. The O and $\widetilde{O}$ are uniform in n, and all of the algorithms here can take ζ as an input rather than fixing it. There are some divisions by powers of ζ, but divisions are included in the definition of algebraic operations.

Pollard uses the special case $h_S(1) = \prod_{s=1}^{\ell}(1 - \zeta^s)$. This is $(1-\zeta)^\ell$ times the quantity $(1+\zeta)(1+\zeta+\zeta^2)\cdots(1+\zeta+\cdots+\zeta^{\ell-1})$. It would be standard to call the latter quantity a "q-factorial" if the letter "q" were used in place of "ζ"; beware, however, that it is not standard to call this quantity a "ζ-factorial". For a vast generalization of Pollard's algorithm to q-holonomic sequences, see [11]; in §4, we will generalize it in a different direction.

3.2. ***An algorithm for the multiplicative version of modular factorials.*** Compute $b = \lfloor\sqrt{\ell}\rfloor$, and define $I = \{1, 2, 3, \ldots, b\}$. Use a product tree to compute the polynomial $h_I(X) = \prod_{i=1}^{b}(X - \zeta^i)$.

Define $J = \{0, b, 2b, \ldots, (b-1)b\}$, and use a remainder tree to compute $h_I(\alpha/\zeta^j)$ for all $j \in J$. Pollard uses the chirp-z transform [50] (Bluestein's trick) instead of a remainder tree, saving a logarithmic factor in the number of operations, and it is also easy to save a logarithmic factor in computing $h_I(X)$, but these speedups are not visible at the level of detail of the analysis in this section.

Multiply ζ^{jb} by $h_I(\alpha/\zeta^j)$ to obtain $\prod_{i=1}^{b}(\alpha - \zeta^{i+j})$ for each j, and then multiply across $j \in J$ to obtain $\prod_{s=1}^{b^2}(\alpha - \zeta^s)$. Finally, multiply by $\prod_{s=b^2+1}^{\ell}(\alpha - \zeta^s)$ to obtain the desired $h_S(\alpha)$.

One can view the product $\prod_{s=1}^{b^2}(\alpha-\zeta^s)$ here, like the product $(b^2)!$ in §2, as the resultant of two degree-b polynomials. Specifically, $\prod_j h_I(\alpha/\zeta^j)$ is the resultant of $\prod_j(X-\alpha/\zeta^j)$ and h_I; and $\prod_j \zeta^{jb} h_I(\alpha/\zeta^j)$ is the resultant of $\prod_j(\zeta^j X-\alpha)$ and h_I. One can, if desired, use continued-fraction resultant algorithms rather than multipoint evaluation via a remainder tree.

3.3. ***Structures in S and f.*** We highlight two structures exploited in the above computation of $\prod_{s=1}^{\ell}(\alpha-\zeta^s)$. First, the set $S = \{1, 2, \ldots, \ell\}$ has enough additive structure to allow most of it to be decomposed as $I + J$, where I and J are much smaller sets. Second, the map $s \mapsto \zeta^s$ is a group homomorphism, allowing each ζ^{i+j} to be computed as the product of ζ^i and ζ^j; we will return to this point in §4.1.

We now formalize the statement regarding additive structure, focusing on the $\mathbb{F}_q$ case that we will need later in the paper. First, some terminology: we say that sets of integers I and J have *no common differences* if $i_1 - i_2 \neq j_1 - j_2$ for all $i_1 \neq i_2$ in I and all $j_1 \neq j_2$ in J. If I and J have no common differences, then the map $I \times J \to I + J$ sending (i, j) to $i + j$ is a bijection.

Lemma 3.4. *Let q be a prime power. Let ζ be an element of $\mathbb{F}_q^*$. Define $h_S(X) = \prod_{s\in S}(X - \zeta^s) \in \mathbb{F}_q[X]$ for each finite subset S of $\mathbb{Z}$. Let I and J be finite subsets of $\mathbb{Z}$ with no common differences. Then*

$$h_{I+J}(X) = \mathrm{Res}_Z(h_I(Z), H_J(X, Z))$$

where $\mathrm{Res}_Z(\cdot,\cdot)$ *is the bivariate resultant, and*

$$H_J(X, Z) := \prod_{j\in J}(X - \zeta^j Z).$$

Proof. $\mathrm{Res}_Z(h_I(Z), H_J(X, Z)) = \prod_{i\in I}\prod_{j\in J}(X - \zeta^i\zeta^j) = \prod_{(i,j)\in I\times J}(X - \zeta^{i+j}) = h_{I+J}(X)$ since the map $I \times J \to I + J$ sending (i, j) to $i + j$ is bijective. □

Algorithm 1 is an algebraic algorithm that outputs $h_S(\alpha)$ given α. The algorithm is parameterized by ζ and the set S, and also by finite subsets $I, J \subset \mathbb{Z}$ with no common differences such that $I + J \subseteq S$. The algorithm and the proof of Proposition 3.5 are stated using generic resultant computation (via continued fractions), but, as in §2.3 and §3.2, one can alternatively use multipoint evaluation.

Proposition 3.5. *Let q be a prime power. Let ζ be an element of $\mathbb{F}_q^*$. Let I, J be finite subsets of $\mathbb{Z}$ with no common differences. Let K be a finite subset of $\mathbb{Z}$ disjoint from $I + J$. Given α in $\mathbb{F}_q$, Algorithm 1 outputs $\prod_{s\in S}(\alpha - \zeta^s)$ using $\widetilde{O}(\max(\#I, \#J, \#K))$ $\mathbb{F}_q$-operations, where $S = (I + J) \cup K$.*

Algorithm 1: Computing $h_S(\alpha) = \prod_{s \in S}(\alpha - \zeta^s)$

Parameters: a prime power q; $\zeta \in \mathbb{F}_q^*$; finite subsets $I, J, K \subseteq \mathbb{Z}$ such that I and J have no common differences and $(I + J) \cap K = \{\}$; ζ^s for each $s \in I \cup J \cup K$

Input: α in $\mathbb{F}_q$

Output: $h_S(\alpha)$ where $h_S(X) = \prod_{s \in S}(X - \zeta^s)$ and $S = (I + J) \cup K$

1 $h_I \leftarrow \prod_{i \in I}(Z - \zeta^i) \in \mathbb{F}_q[Z]$

2 $H_J \leftarrow \prod_{j \in J}(\alpha - \zeta^j Z) \in \mathbb{F}_q[Z]$

3 $h_{I+J} \leftarrow \mathrm{Res}_Z(h_I, H_J) \in \mathbb{F}_q$

4 $h_K \leftarrow \prod_{k \in K}(\alpha - \zeta^k) \in \mathbb{F}_q$

5 **return** $h_{I+J} \cdot h_K$

The $\widetilde{O}$ is uniform in q. Instead of taking ζ and various precomputed powers of ζ as parameters, the algorithm can take ζ as an input, at the cost of computing ζ^i for $i \in I$, ζ^j for $j \in J$, and ζ^k for $k \in K$. This preserves the time bound if the elements of I, J, K each have polylog$(\max(\#I, \#J, \#K))$ bits.

Proof. Since $S \setminus K = I + J$, we have $h_S(\alpha) = h_{I+J}(\alpha) \cdot h_K(\alpha)$, and Lemma 3.4 shows that $h_{I+J}(\alpha) = \mathrm{Res}_Z(h_I(Z), H_J(\alpha, Z))$. Line 1 computes $h_I(Z)$ in $\widetilde{O}(\#I)$ $\mathbb{F}_q$-operations; Line 2 computes $H_J(\alpha, Z)$ in $\widetilde{O}(\#J)$ $\mathbb{F}_q$-operations; Line 3 computes $h_{I+J}(\alpha)$ in $\widetilde{O}(\max(\#I, \#J))$ $\mathbb{F}_q$-operations; and Line 4 computes $h_K(\alpha)$ in $\widetilde{O}(\#K)$ $\mathbb{F}_q$-operations. The total is $\widetilde{O}(\max(\#I, \#J, \#K))$ $\mathbb{F}_q$-operations. □

3.6. ***Optimization.*** The best conceivable case for the time bound in Proposition 3.5, as a function of $\#S$, is $\widetilde{O}(\sqrt{\#S})$. Indeed, $\#S = \#I \cdot \#J + \#K$, so $\max(\#I, \#J, \#K) \geq \sqrt{\#S + 1/4} - 1/2$.

To reach $\widetilde{O}(\sqrt{\#S})$ for a given set of exponents S, we need sets I and J with no common differences such that $I + J \subseteq S$ with $\#I$, $\#J$, and $\#(S \setminus (I + J))$ in $\widetilde{O}(\sqrt{\#S})$. Such I and J exist for many useful sets S. Example 3.7 shows a simple form for I and J when S is an arithmetic progression.

Example 3.7. Suppose S is an arithmetic progression of length n: that is,

$$S = \{m, m + r, m + 2r, \ldots, m + (n - 1)r\}$$

for some m and some nonzero r. Let $b = \lfloor \sqrt{n} \rfloor$, and set

$$I := \{ir \mid 0 \leq i < b\} \qquad \text{and} \qquad J := \{m + jbr \mid 0 \leq j < b\};$$

then I and J have no common differences, and $I + J = \{m + kr \mid 0 \leq k < b^2\}$, so

$$I + J = S \setminus K \quad \text{where} \quad K = \{m + kr \mid b^2 \leq k < n\}.$$

Now $\#I = \#J = b$, and $\#K = n - b^2 \leq 2b$, so we can use these sets to compute $h_S(\alpha)$ in $\widetilde{O}(b) = \widetilde{O}(\sqrt{n})$ $\mathbb{F}_q$-operations, following Proposition 3.5. (In the case $r = 1$, we recognise the index sets driving Shanks' baby-step giant-step algorithm.)

4. Elliptic resultants

The technique in §3 for evaluating polynomials whose roots are powers is well known. Our main theoretical contribution is to adapt this to polynomials whose roots are functions of more interesting groups: in particular, functions of elliptic-curve torsion points. The most important such function is the x-coordinate. The main complication here is that, unlike in §3, the function x is not a homomorphism.

4.1. ***The elliptic setting.*** Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve, let $P \in \mathcal{E}(\mathbb{F}_q)$, and define $\mathcal{G} = \langle P \rangle$. Let S be a finite subset of $\mathbb{Z}$. We want to evaluate

$$h_S(X) = \prod_{s \in S} (X - f([s]P)), \qquad \text{where} \qquad f : Q \longmapsto \begin{cases} 0 & \text{if } Q = 0, \\ x(Q) & \text{if } Q \neq 0, \end{cases}$$

at some α in $\mathbb{F}_q$. Here $x : \mathcal{E} \to \mathcal{E}/\langle \pm 1 \rangle \cong \mathbb{P}^1$ is the usual map to the x-line.

Adapting Algorithm 1 to this setting is not a simple matter of replacing the multiplicative group with an elliptic curve. Indeed, Algorithm 1 explicitly uses the homomorphic nature of $f : s \mapsto \zeta^s$ to represent the roots ζ^s as $\zeta^i \zeta^j$ where $s = i + j$. This presents an obstacle when moving to elliptic curves: $x([i+j]P)$ is not a rational function of $x([i]P)$ and $x([j]P)$, so we cannot apply the same trick of decomposing most of S as $I + J$ before taking a resultant of polynomials encoding $f(I)$ and $f(J)$.

This obstacle does not matter in the factorization context. For example, in §3, a straightforward resultant $\prod_{i,j}(\alpha/\zeta^j - \zeta^i)$ detects collisions between α/ζ^j and ζ^i; our rescaling to $\prod_{i,j}(\alpha - \zeta^{i+j})$ was unnecessary. Similarly, Montgomery's FFT extension [44] to ECM computes a straightforward resultant $\prod_{i,j}(x([i]P) - x([j]P))$, detecting any collisions between $x([i]P)$ and $x([j]P)$; this factorization method does not compute, and does not need to compute, a product of functions of $x([i + j]P)$. The isogenies context is different: we need a product of functions of $x([i + j]P)$.

Fortunately, even if the x-map is not homomorphic, there is an algebraic relation between $x(P)$, $x(Q)$, $x(P + Q)$, and $x(P - Q)$, which we will review in §4.2. The introduction of the difference $x(P - Q)$ as well as the sum $x(P + Q)$ requires us to replace the decomposition of most of S as $I + J$ with a decomposition involving $I + J$ and $I - J$, which we will formalize in §4.5. We define the resultant required to tie all this together and compute $h_{I \pm J}(\alpha)$ in §4.8.

4.2. ***Biquadratic relations on x-coordinates.*** Lemma 4.3 recalls the general relationship between $x(P)$, $x(Q)$, $x(P + Q)$, and $x(P - Q)$. Example 4.4 gives explicit formulæ for the case that is most useful in our applications.

Lemma 4.3. *Let q be a prime power. Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve. There exist biquadratic polynomials F_0, F_1, and F_2 in $\mathbb{F}_q[X_1, X_2]$ such that*

$$(X - x(P + Q))(X - x(P - Q)) = X^2 + \frac{F_1(x(P), x(Q))}{F_0(x(P), x(Q))} X + \frac{F_2(x(P), x(Q))}{F_0(x(P), x(Q))}$$

for all P and Q in $\mathcal{E}$ such that $0 \notin \{P, Q, P + Q, P - Q\}$.

Proof. The existence of F_0, F_1, and F_2 is classical (see e.g. [17, p. 132] for the F_i for Weierstrass models); indeed, the existence of such biquadratic systems is a general phenomenon for theta functions of level 2 on abelian varieties (see e.g. [47, §3]). □

Example 4.4 (biquadratics for Montgomery models). If $\mathcal{E}$ is defined by an affine equation $By^2 = x(x^2 + Ax + 1)$, then the polynomials of Lemma 4.3 are

$$\begin{aligned} F_0(X_1, X_2) &= (X_1 - X_2)^2\,, \\ F_1(X_1, X_2) &= -2((X_1X_2 + 1)(X_1 + X_2) + 2AX_1X_2)\,, \\ F_2(X_1, X_2) &= (X_1X_2 - 1)^2\,. \end{aligned}$$

The symmetric triquadratic polynomial $(X_0X_1 - 1)^2 + (X_0X_2 - 1)^2 + (X_1X_2 - 1)^2 - 2X_0X_1X_2(X_0 + X_1 + X_2 + 2A) - 2$ is $X_0^2F_0(X_1, X_2) + X_0F_1(X_1, X_2) + F_2(X_1, X_2)$.

Montgomery curves $By^2 = x(x^2+Ax+1)$, and the remarkably simple formula $(X_1X_2-1)^2/(X_1-X_2)^2$ for the product $x(P + Q)x(P - Q)$ on these curves, were introduced by Montgomery in [43, Section 10.3.1]. See [7] for more information about Montgomery curves.

4.5. ***Index systems.*** In §3, we represented most of S as $I + J$; requiring I and J to have no common differences ensured this representation had no redundancy. Now we will represent most elements of S as elements of $(I + J) \cup (I - J)$, so we need a stronger restriction on I and J to avoid redundancy.

Definition 4.6. Let I and J be finite sets of integers.

(1) We say that (I, J) is an *index system* if the maps $I \times J \to \mathbb{Z}$ defined by $(i, j) \mapsto i + j$ and $(i, j) \mapsto i - j$ are both injective and have disjoint images.

(2) If S is a finite subset of $\mathbb{Z}$, then we say that an index system (I, J) is an *index system for* S if $I + J$ and $I - J$ are both contained in S.

If (I, J) is an index system, then the sets $I + J$ and $I - J$ are both in bijection with $I \times J$. We write $I \pm J$ for the union of $I + J$ and $I - J$.

Example 4.7. Let m be an odd positive integer, and consider the set

$$S = \{1, 3, 5, \ldots, m\}$$

in arithmetic progression. Let

$$I := \{2b(2i + 1) \mid 0 \le i < b'\} \qquad \text{and} \qquad J := \{2j + 1 \mid 0 \le j < b\}$$

where $b = \lfloor \sqrt{m + 1}/2 \rfloor$; $b' = \lfloor (m + 1)/4b \rfloor$ if $b > 0$; and $b' = 0$ if $b = 0$. Then (I, J) is an index system for S, and $S \setminus (I \pm J) = K$ where $K = \{4bb' + 1, \ldots, m - 2, m\}$. If $b > 0$ then $\#I = b' \le b + 2$, $\#J = b$, and $\#K \le 2b - 1$.

4.8. ***Elliptic resultants.*** We are now ready to adapt the results of §3 to the setting of §4.1. Our main tool is Lemma 4.9, which expresses $h_{I\pm J}$ as a resultant of smaller polynomials.

Lemma 4.9. *Let q be a prime power. Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve. Let P be an element of $\mathcal{E}(\mathbb{F}_q)$. Let n be the order of P. Let (I, J) be an index system such that I, J, $I+J$, and $I-J$ do not contain any elements of $n\mathbb{Z}$. Then*

$$h_{I\pm J}(X) = \frac{1}{\Delta_{I,J}} \cdot \mathrm{Res}_Z\left(h_I(Z), E_J(X, Z)\right)$$

where

$$E_J(X, Z) := \prod_{j\in J} \left(F_0(Z, x([j]P))X^2 + F_1(Z, x([j]P))X + F_2(Z, x([j]P))\right)$$

and $\Delta_{I,J} := \mathrm{Res}_Z\left(h_I(Z), D_J(Z)\right)$ where $D_J(Z) := \prod_{j\in J} F_0(Z, x([j]P))$.

Proof. Since (I, J) is an index system, $I+J$ and $I-J$ are disjoint, and therefore we have $h_{I\pm J}(X) = h_{I+J}(X) \cdot h_{I-J}(X)$. Expanding and regrouping terms, we get

$$\begin{aligned} h_{I\pm J}(X) &= \prod_{(i,j)\in I\times J} (X - x([i+j]P))\,(X - x([i-j]P)) \\ &= \prod_{i\in I}\prod_{j\in J}\left(X^2 + \frac{F_1(x([i]P), x([j]P))}{F_0(x([i]P), x([j]P))}X + \frac{F_2(x([i]P), x([j]P))}{F_0(x([i]P), x([j]P))}\right) \end{aligned}$$

by Lemma 4.3. Factoring out the denominator, we find

$$h_{I\pm J}(X) = \frac{\prod_{i\in I} E_J(X, x([i]P))}{\prod_{i\in I}\prod_{j\in J} F_0(x([i]P), x([j]P))} = \frac{\prod_{i\in I} E_J(X, x([i]P))}{\prod_{i\in I} D_J(x([i]P))};$$

and finally $\prod_{i\in I} E_J(X, x([i]P)) = \mathrm{Res}_Z(h_I(Z), E_J(X, Z))$ and $\prod_{i\in I} D_J(x([i]P)) = \mathrm{Res}_Z(h_I(Z), D_J(Z)) = \Delta_{I,J}$, which yields the result. □

4.10. ***Elliptic polynomial evaluation.*** Algorithm 2 is an algebraic algorithm for computing $h_S(\alpha)$; it is the elliptic analogue of Algorithm 1. Theorem 4.11 proves its correctness and runtime.

Theorem 4.11. *Let q be a prime power. Let $\mathcal{E}/\mathbb{F}_q$ be an elliptic curve. Let P be an element of $\mathcal{E}(\mathbb{F}_q)$. Let n be the order of P. Let (I, J) be an index system for a finite set $S \subset \mathbb{Z}$. Assume that I, J, and S contain no elements of $n\mathbb{Z}$. Given α in $\mathbb{F}_q$, Algorithm 2 computes*

$$h_S(\alpha) = \prod_{s\in S}\left(\alpha - x([s]P)\right)$$

in $\widetilde{O}(\max(\#I, \#J, \#K))$ $\mathbb{F}_q$-operations, where $K = S \setminus (I \pm J)$.

In particular, if $\#I$, $\#J$, and $\#K$ are in $\widetilde{O}(\sqrt{\#S})$, then Algorithm 2 computes $h_S(\alpha)$ in $\widetilde{O}(\sqrt{\#S})$ $\mathbb{F}_q$-operations. The $\widetilde{O}$ is uniform in q. Instead of taking P and various $x([s]P)$ as parameters, Algorithm 2 can take P as an input, at the cost of computing the relevant multiples of P.

Algorithm 2: Computing $h_S(\alpha) = \prod_{s\in S}\left(\alpha - x([s]P)\right)$ for $P \in \mathcal{E}(\mathbb{F}_q)$

Parameters: a prime power q; an elliptic curve $\mathcal{E}/\mathbb{F}_q$; $P \in \mathcal{E}(\mathbb{F}_q)$; a finite subset $S \subset \mathbb{Z}$; an index system (I, J) for S such that $S \cap n\mathbb{Z} = I \cap n\mathbb{Z} = J \cap n\mathbb{Z} = \{\}$, where n is the order of P; $x([s]P)$ for each $s \in I \cup J \cup K$

Input: α in $\mathbb{F}_q$

Output: $h_S(\alpha)$ where $h_S(X) = \prod_{s\in S}(X - x([s]P))$

1 $h_I \leftarrow \prod_{i\in I}(Z - x([i]P)) \in \mathbb{F}_q[Z]$

2 $D_J \leftarrow \prod_{j\in J} F_0(Z, x([j]P)) \in \mathbb{F}_q[Z]$

3 $\Delta_{I,J} \leftarrow \mathrm{Res}_Z(h_I, D_J) \in \mathbb{F}_q$

4 $E_J \leftarrow \prod_{j\in J}\left(F_0(Z, x([j]P))\alpha^2 + F_1(Z, x([j]P))\alpha + F_2(Z, x([j]P))\right) \in \mathbb{F}_q[Z]$

5 $R \leftarrow \mathrm{Res}_Z(h_I, E_J) \in \mathbb{F}_q$

6 $h_K \leftarrow \prod_{k\in S\setminus(I\pm J)}(\alpha - x([k]P)) \in \mathbb{F}_q$

7 **return** $h_K \cdot R/\Delta_{I,J}$

Proof. The proof follows that of Proposition 3.5. Since $S \setminus K = I \pm J$, we have $h_S(\alpha) = h_{I\pm J}(\alpha) \cdot h_K(\alpha)$. Using the notation of Lemma 4.9: Line 1 computes $h_I(Z)$ in $\widetilde{O}(\#I)$ $\mathbb{F}_q$-operations; Line 2 computes $D_J(Z)$ in $\widetilde{O}(\#J)$ $\mathbb{F}_q$-operations; Line 3 computes $\Delta_{I,J}$ in $\widetilde{O}(\max(\#I, \#J))$ $\mathbb{F}_q$-operations; Line 4 computes $E_J(\alpha, Z)$ in $\widetilde{O}(\#J)$ $\mathbb{F}_q$-operations; Line 5 computes $\mathrm{Res}_Z(h_I(Z), E_J(\alpha, Z))$, which is the same as $\Delta_{I,J}h_{I\pm J}(\alpha)$ by Lemma 4.9, in $\widetilde{O}(\max(\#I, \#J))$ $\mathbb{F}_q$-operations; Line 6 computes $h_K(\alpha)$ in $\widetilde{O}(\#K)$ $\mathbb{F}_q$-operations; and Line 7 returns $h_S(\alpha) = h_K(\alpha) \cdot h_{I\pm J}(\alpha)$. The total number of $\mathbb{F}_q$-operations is in $\widetilde{O}(\max(\#I, \#J, \#K))$. □

Example 4.12 (evaluating kernel polynomials). We now address a problem from the introduction: evaluating $\Psi_{\mathcal{G}}$, the radical of the denominators of the rational functions defining the ℓ-isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ with kernel $\mathcal{G} = \langle P\rangle$, for ℓ odd. Here

$$\Psi_{\mathcal{G}}(X) = h_S(X) = \prod_{s\in S}(X - x([s]P)) \qquad \text{where} \qquad S = \{1, 3, \ldots, \ell - 2\}$$

(the set S may be replaced by any set of representatives of $((\mathbb{Z}/\ell\mathbb{Z}) \setminus \{0\})/\langle\pm 1\rangle$). Following Example 4.7, let $I = \{2b(2i + 1) \mid 0 \le i < b'\}$ and $J = \{1, 3, \ldots, 2b - 1\}$ with $b = \lfloor\sqrt{\ell - 1}/2\rfloor$ and (for $b > 0$) $b' = \lfloor(\ell - 1)/4b\rfloor$; then (I, J) is an index system for S, and Algorithm 2 computes $h_S(\alpha) = \Psi_{\mathcal{G}}(\alpha)$ for any α in $\mathbb{F}_q$ in $\widetilde{O}(\sqrt{\ell})$ $\mathbb{F}_q$-operations.

Example 4.13 (evaluating derivatives of polynomials). Algorithm 2 can evaluate h_S at points in any $\mathbb{F}_q$-algebra, at the cost of a slowdown that depends on how large the algebra is. These algebras need not be fields. For example, we can evaluate $h_S(\alpha+\epsilon)$ in the algebra $\mathbb{F}_q[\epsilon]/\epsilon^2$ of 1-jets, obtaining $h_S(\alpha)+\epsilon h'_S(\alpha)$. We can thus evaluate derivatives, sums over roots, etc. The algebra of 1-jets was used the same way in, e.g., [46; 40; 5]; [2] also notes Zagier's suggested terminology "jet plane".

4.14. ***Irrational generators.*** The point P in Lemma 4.9, Algorithm 2, and Theorem 4.11 need not be in $\mathcal{E}(\mathbb{F}_q)$: everything is defined over $\mathbb{F}_q$ if $x(P)$ is in $\mathbb{F}_q$. More generally, take P in $\mathcal{E}(\mathbb{F}_{q^e})$ with $x(P)$ in $\mathbb{F}_{q^e}$ for some minimal $e \geq 1$. The q-power Frobenius π on $\mathcal{E}$ maps P to $\pi(P) = [\lambda]P$ for some eigenvalue λ in $\mathbb{Z}/n\mathbb{Z}$ of order e in $(\mathbb{Z}/n\mathbb{Z})^*$. Let $L = \{\lambda^a \mid 0 \leq a < e\}$. For $h_S(X)$ to be in $\mathbb{F}_q[X]$, we need $S = LS'$ for some $S' \subseteq \mathbb{Z}$ (modulo n): that is, $S = \{\lambda^a s' \mid 0 \leq a < e, s' \in S'\}$. Then

$$h_S(X) = \prod_{s' \in S'} \prod_{a=0}^{e-1} (X - x([\lambda^a s']P)) = \prod_{s' \in S'} g_{s'}(X)$$

where the polynomial

$$g_{s'}(X) = \prod_{a=0}^{e-1} (X - x([\lambda^a s']P)) = \prod_{a=0}^{e-1} (X - x(\pi^a([s']P))) = \prod_{a=0}^{e-1} (X - x([s']P)^{q^a})$$

is in $\mathbb{F}_q[X]$, and can be easily computed from $x([s]P)$.

To write h_I, D_J, and E_J as products of polynomials over $\mathbb{F}_q$, we need the index system (I, J) for S to satisfy $(I, J) = (LI', LJ')$ for some index system (I', J') for S'. While this does not affect the asymptotic complexity of the resulting evaluation algorithms at our level of analysis, it should be noted that the requirement that $(I, J) = (LI', LJ')$ is quite strong: typically e is in $O(\ell)$, so $\#L$ is not in $\widetilde{O}(\sqrt{\#S})$, and a suitable index system (I, J) with $\#I$ and $\#J$ in $\widetilde{O}(\sqrt{\#S})$ does not exist.

4.15. ***Other functions on $\mathcal{E}$.*** We can replace x with more general functions on $\mathcal{E}$, though for completely general f there may be no useful analogue of Lemma 4.3, or at least not one that allows a Lemma 4.9 with conveniently small index system. However, everything above adapts easily to the case where x is composed with an automorphism of $\mathbb{P}^1$ (that is, $f = (ax + b)/(cx + d)$ with a, b, c, d in $\mathbb{F}_q$ such that $ad \neq bc$). Less trivially, we can take $f = \psi_x$ for any isogeny $\psi : \mathcal{E} \to \mathcal{E}''$. In this case, the F_0, F_1, and F_2 of Lemma 4.3 are derived from the curve $\mathcal{E}''$, not $\mathcal{E}$.

4.16. ***Abelian varieties.*** It is tempting to extend our results to higher-dimensional principally polarized abelian varieties (PPAVs), replacing $\mathcal{E}$ with a PPAV $\mathcal{A}/\mathbb{F}_q$, and x with some coordinate on $\mathcal{A}$, but evaluating the resulting h_S using our methods is more complicated. The main issue is the analogue of Lemma 4.3. If we choose any even coordinate x on $\mathcal{A}$, then the classical theory of theta functions yields quadratic relations between $x(P + Q)$, $x(P - Q)$, and the coordinates of P and Q, but not *only* $x(P)$ and $x(Q)$: they also require the other even coordinates of P and Q. (The simplest example of this is seen in the differential addition formulæ for Kummer surfaces: see [22, §6], [31, §3.2], and [18, §4.4].) This means that an analogue of Algorithm 2 for PPAVs would require multivariate polynomials and resultants; an investigation of this is well beyond the scope of this article.

5. Computing elliptic isogenies

We now apply the techniques of §4 to the problem of efficient isogeny computation. The task is divided in two parts: evaluating isogenies on points (§5.1), and computing codomain curves (§5.2). Our cryptographic applications use isogenies between Montgomery models of elliptic curves, and we concentrate exclusively on this case here; but our methods adapt easily to Weierstrass and other models.

5.1. ***Evaluating isogenies.*** Let $\mathcal{E}/\mathbb{F}_q : y^2 = x(x^2 + Ax + 1)$ be an elliptic curve in Montgomery form, and let P be a point of prime order $\ell \neq 2$ in $\mathcal{E}(\mathbb{F}_q)$. Costello and Hisil give explicit formulæ in [25] for a quotient isogeny $\phi : \mathcal{E} \to \mathcal{E}'$ with kernel $\mathcal{G} = \langle P \rangle$ such that $\mathcal{E}'/\mathbb{F}_q : y^2 = x(x^2 + A'x + 1)$ is a Montgomery curve:

$$\phi : (X, Y) \longmapsto \big(\phi_x(X), c_0 Y \phi'_x(X)\big)$$

where $c_0 = \prod_{0<s<\ell/2} x([s]P)$ and

$$\phi_x(X) = X \prod_{0<s<\ell} \frac{x([s]P)X - 1}{X - x([s]P)}\,. \tag{1}$$

See [51] for generalizations and a different proof, and see the earlier paper [45] for analogous Edwards-coordinate formulas.

Our main goal is to evaluate ϕ on the level of x-coordinates: that is, to compute $\phi_x(\alpha)$ given $\alpha = x(Q)$ for Q in $\mathcal{E}(\mathbb{F}_q)$. This is sufficient for our cryptographic applications. Applications that also need the y-coordinate of $\phi(Q)$, namely $c_0 y(Q)\phi'_x(\alpha)$, can compute c_0 as $(-1)^{(\ell-1)/2} h_S(0)$, and can compute $\phi'_x(\alpha)$ together with $\phi_x(\alpha)$ by the technique of Example 4.13. To compute $\phi_x(\alpha)$, we rewrite Eq. (1) as

$$\phi_x(X) = \frac{X^\ell \cdot h_S(1/X)^2}{h_S(X)^2} \quad \text{where} \quad S = \{1, 3, \ldots, \ell - 2\}\,.$$

Computing $\phi_x(\alpha)$ thus reduces to two applications of Algorithm 4.11, using (for example) the index system (I, J) for S in Example 4.7. The constant $\Delta_{I,J}$ appears with the same multiplicity in the numerator and denominator, so we need not compute it. All divisions in the computation are by nonzero field elements except in the following cases, which can be handled separately: if $Q = 0$ then $\phi(Q) = 0$; if $Q \neq 0$ but $h_S(\alpha) = 0$ for $\alpha = x(Q)$ then $\phi(Q) = 0$; if $Q = (0, 0)$ then $\phi(Q) = (0, 0)$.

5.2. ***Computing codomain curves.*** Our other main task is to determine the coefficient A' in the defining equation of $\mathcal{E}'$.

One approach is as follows. We can now efficiently compute $\phi(Q)$ for any Q in $\mathcal{E}(\mathbb{F}_q)$. Changing the base ring from $\mathbb{F}_q$ to $R = \mathbb{F}_q[\alpha]/(\alpha^2 + A\alpha + 1)$ (losing a small constant factor in the cost of evaluation) gives us $\phi(Q)$ for any Q in $\mathcal{E}(R)$. In particular, $Q = (\alpha, 0)$ is a point in $\mathcal{E}[2](R)$, and computing $\phi(Q) = (\alpha', 0)$ reveals $A' = -(\alpha' + 1/\alpha')$. An alternative—at the expense of taking a square root, which is no longer a q-independent algebraic computation—is to find a point $(\alpha, 0)$ in $\mathcal{E}(\mathbb{F}_{q^2})$ with $\alpha \neq 0$. Sometimes α is in $\mathbb{F}_q$, and then extending to $\mathbb{F}_{q^2}$ is unnecessary.

Another approach is to use explicit formulas for A'. The formulas from [25] give $A' = c_0^2(A - 3\sigma)$ where $c_0^2 = \prod_{0<s<\ell} x([s]P)$ and $\sigma = \sum_{0<s<\ell}(x([s]P) - 1/x([s]P))$. As pointed out in [42] in the context of CSIDH, one can instead transform to twisted Edwards form and use the formulas from [45], obtaining $A' = 2(1+d)/(1-d)$ where

$$d = \left(\frac{A-2}{A+2}\right)^{\ell} \left(\prod_{s\in S} \frac{x([s]P)-1}{x([s]P)+1}\right)^{8} = \left(\frac{A-2}{A+2}\right)^{\ell} \left(\frac{h_S(1)}{h_S(-1)}\right)^{8}.$$

We can thus compute A' using $\widetilde{O}(\sqrt{\ell})$ operations: every task we need can be performed by some evaluations of h_S and some (asymptotically negligible) operations.

6. Applications in isogeny-based cryptography

With the notable exception of SIDH/SIKE [36; 27; 1], most isogeny-based cryptographic protocols need to evaluate large-degree isogenies. Specifically, CRS [52; 26], CSIDH [20], CSURF [19], etc. use large-degree isogenies, since not enough keys are fast compositions of isogenies of a few small prime degrees. The largest isogeny degree, with standard optimizations, grows quasi-linearly in the pre-quantum security level. For the same post-quantum security level, known quantum attacks require an asymptotically larger base field but do not affect the largest isogeny degree; see [20, Remark 11].

Concretely, targeting 128 bits of pre-quantum security, CSIDH-512 fixes

$$p = 4 \cdot \underbrace{(3 \cdot 5 \cdots 373)}_{\text{73 first odd primes}} \cdot 587 - 1$$

and uses isogenies of all odd prime degrees $\ell \mid p+1$. Similarly, CSURF-512 fixes

$$p = 8 \cdot 9 \cdot \underbrace{(5 \cdot 7 \cdots 337)}_{\text{66 consecutive primes}} \cdot 349 \cdot 353 \underbrace{(367 \cdots 389)}_{\text{6 consecutive primes}} - 1$$

and uses isogenies of all prime degrees $\ell \mid p+1$, including $\ell = 2$.

The CSIDH and CSURF algorithms repeatedly sample a random point of order dividing $p+1$ in $\mathcal{E}/\mathbb{F}_p$, multiply it by an appropriate cofactor to get P, and then apply Vélu's formulas for each of the primes $\ell \mid \mathrm{ord}(P)$ to obtain $\mathcal{E}' = \mathcal{E}/\langle P\rangle$. Our algorithm seamlessly replaces Vélu's formulas in both systems. Computing $\mathcal{E}'$ is easy in CSURF: all curves involved have rational 2-torsion, and can thus be represented by a root of $\alpha^2 + A\alpha - 1$ in $\mathbb{F}_p$. For CSIDH, we can apply the techniques of §5.2; alternatively, we can walk to the surface and represent curves as in CSURF.

B-SIDH [24] is an SIDH variant using smaller prime fields, at the cost of much larger prime isogeny degrees. One participant uses isogenies of degree $\ell \mid p+1$, and the other uses $\ell \mid p-1$. Since primes p such that $p-1$ and $p+1$ *both* have many small prime factors are rare, some of the ℓ involved in B-SIDH tend to be even larger than in CSIDH and CSURF. The B-SIDH algorithm starts from a single point P and computes $\mathcal{E}/\langle P\rangle$ together with the evaluation of $\phi : \mathcal{E} \to \mathcal{E}/\langle P\rangle$ at three points. Unlike CSIDH and

CSURF, there is no repeated random sampling of points: a single ℓ-isogeny evaluation for each prime $\ell \mid p \pm 1$ is needed.

Our asymptotic speedup in isogeny evaluation implies asymptotic speedups for CRS, CSIDH, CSURF, and B-SIDH as the security level increases. This does not imply, however, that there is a speedup for (e.g.) pre-quantum security 2^{128}.

The Appendix of this paper's full version [6] addresses the question of how large ℓ needs to be before our algorithms become faster than the conventional algorithms. It looks more closely at performance and quantifies the cross-over point by considering different metrics such as time in several software or the number of multiplications. More precisely, we present four implementations: one in magma [10], one in julia [9] (with nemo [29] for the underlying arithmetic) and two in C (a first one using the underlying arithmetic of FLINT [32] and the second one on top of the arithmetic subroutines of [20]). In each of the metrics considered there, the cross-over point is within the range of primes used in CSIDH-512. The new ℓ-isogeny algorithm sets new speed records for CSIDH-512 and CSIDH-1024 by small but measurable percentages, and has more effect on protocols that use larger ℓ-isogenies. Our code is available from https://velusqrt.isogeny.org.

Cryptographic protocols that exploit the KLPT algorithm [37] for isogeny path renormalization, such as the signature scheme [30] and the encryption scheme SÉTA [28], need to work with irrational torsion points. They may thus benefit from the technique of §4.14. We did not investigate these protocols further.

References

[1] Reza Azarderakhsh, Brian Koziel, Matt Campagna, Brian LaMacchia, Craig Costello, Patrick Longa, Luca De Feo, Michael Naehrig, Basil Hess, Joost Renes, Amir Jalali, Vladimir Soukharev, David Jao, and David Urbanik. Supersingular isogeny key encapsulation, 2017. https://sike.org.

[2] Karim Belabas, Hendrik W. Lenstra, Jr., and Don B. Zagier. Explicit methods in number theory, 2011. Oberwolfach report 35/2011. https://publications.mfo.de/handle/mfo/3250.

[3] Daniel J. Bernstein. Scaled remainder trees, 2004. https://cr.yp.to/papers.html#scaledmod.

[4] Daniel J. Bernstein. Reducing lattice bases to find small-height values of univariate polynomials. In Joe Buhler and Peter Stevenhagen, editors, *Algorithmic number theory: lattices, number fields, curves and cryptography*, pages 421–446. Cambridge University Press, 2008. https://cr.yp.to/papers.html#smallheight.

[5] Daniel J. Bernstein. Jet list decoding, 2011. https://cr.yp.to/talks/2011.11.24/slides.pdf.

[6] Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. Cryptology ePrint Archive, Report 2020/341, 2020. https://eprint.iacr.org/2020/341.

[7] Daniel J. Bernstein and Tanja Lange. Montgomery curves and the Montgomery ladder. In Joppe W. Bos and Arjen K. Lenstra, editors, *Topics in computational number theory inspired by Peter L. Montgomery*, pages 82–115. Cambridge University Press, 2017. https://eprint.iacr.org/2017/293.

[8] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT*, volume 11477 of *Lecture Notes in Computer Science*, pages 409–441, 2019. https://ia.cr/2018/1059.

[9] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017. https://arxiv.org/abs/1411.1607.

[10] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993). https://www.math.ru.nl/ bosma/pubs/JSC1997Magma.pdf.

[11] Alin Bostan. Computing the N-th term of a q-holonomic sequence. In *Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation*, ISSAC '20, page 46–53, New York, NY, USA, 2020. Association for Computing Machinery.

[12] Alin Bostan, Pierrick Gaudry, and Éric Schost. Linear recurrences with polynomial coefficients and application to integer factorization and Cartier–Manin operator. *SIAM Journal on Computing*, 36(6):1777–1806, 2007. https://hal.inria.fr/inria-00514132.

[13] Alin Bostan, Grégoire Lecerf, Bruno Salvy, Éric Schost, and Bernd Wiebelt. Complexity issues in bivariate polynomial factorization. ISSAC 2004, pages 42–49. Association for Computing Machinery, 2004. https://specfun.inria.fr/bostan/publications/BoLeSaScWi04.pdf.

[14] Alin Bostan, Grégoire Lecerf, and Éric Schost. Tellegen's principle into practice. ISSAC 2003, pages 37–44. Association for Computing Machinery, 2003. https://specfun.inria.fr/bostan/publications/BoLeSc03.pdf.

[15] Richard P. Brent and H. T. Kung. The area-time complexity of binary multiplication. *Journal of the ACM*, 28:521–534, 1981. https://maths-people.anu.edu.au/ brent/pub/pub055.html.

[16] Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.

[17] J. W. S. Cassels. *Lectures on Elliptic Curves*, volume 24 of *London Mathematical Society Student Texts*. Cambridge University Press, 1991.

[18] J. W. S. Cassels and E. V. Flynn. *Prolegomena to a middlebrow arithmetic of curves of genus 2*, volume 230 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1996.

[19] Wouter Castryck and Thomas Decru. CSIDH on the surface. 2019. https://ia.cr/2019/1404.

[20] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *ASIACRYPT (3)*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427, 2018. https://ia.cr/2018/383.

[21] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. Stronger and faster side-channel protections for CSIDH. In *Progress in Cryptology – LATINCRYPT 2019*, pages 173–193, 2019. https://ia.cr/2019/837.

[22] David V. Chudnovsky and Gregory V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics*, 7(4):385–434, 1986. https://core.ac.uk/download/pdf/82012348.pdf.

[23] Edgar Costa and David Harvey. Faster deterministic integer factorization. *Mathematics of Computation*, 83(285):339–345, 2014. https://arxiv.org/abs/1201.2116.

[24] Craig Costello. B-SIDH: supersingular isogeny Diffie-Hellman using twisted torsion, 2019. https://ia.cr/2019/1145.

[25] Craig Costello and Hüseyin Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 303–329, 2017. https://eprint.iacr.org/2017/504.

[26] Jean-Marc Couveignes. Hard homogeneous spaces, 2006. https://ia.cr/2006/291.

[27] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014. https://ia.cr/2011/506.

[28] Cyprien Delpech de Saint Guilhem, Péter Kutas, Christophe Petit, and Javier Silva. SÉTA: supersingular encryption from torsion attacks, 2019. https://ia.cr/2019/1291.

[29] Claus Fieker, William Hart, Tommy Hofmann, and Fredrik Johansson. Nemo/Hecke: Computer algebra and number theory packages for the Julia programming language. ISSAC 2017, pages 157–164, New York, NY, USA, 2017. ACM. https://arxiv.org/abs/1705.06134v1.

[30] Steven D. Galbraith, Christophe Petit, and Javier Silva. Identification protocols and signature schemes based on supersingular isogeny problems. In *ASIACRYPT*, 2017. https://ia.cr/2016/1154.

[31] Pierrick Gaudry. Fast genus 2 arithmetic based on Theta functions. *J. Mathematical Cryptology*, 1(3):243–265, 2007. https://ia.cr/2005/314.

[32] William B. Hart, Fredrik Johansson, and Sebastian Pancratz. FLINT: Fast Library for Number Theory, 2020. Development version, http://flintlib.org.

[33] David Harvey. Faster algorithms for the square root and reciprocal of power series. *Mathematics of Computation*, 80(273): 387–394, 2011. https://arxiv.org/abs/0910.1926.

[34] Markus Hittmeir. A babystep-giantstep method for faster deterministic integer factorization. *Mathematics of Computation*, 87(314):2915–2935, 2018. https://arxiv.org/abs/1608.08766v1.

[35] Aaron Hutchinson, Jason LeGrow, Brian Koziel, and Reza Azarderakhsh. Further optimizations of CSIDH: A systematic approach to efficient strategies, permutations, and bound vectors, 2019. https://ia.cr/2019/1121.

[36] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *PQCrypto 2011*, pages 19–34, 2011. https://ia.cr/2011/506.

[37] David Kohel, Kristin E. Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem. https://ia.cr/2014/505.

[38] David R. Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California at Berkeley, 1996. http://iml.univ-mrs.fr/ kohel/pub/thesis.pdf.

[39] Hendrik W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987. https://www.math.leidenuniv.nl/ hwl/PUBLICATIONS/1987c/art.pdf.

[40] Gregorio Malajovich and Jorge P. Zubelli. Tangent Graeffe iteration. *Numerische Mathematik*, 89(4):749–782, 2001. https://arxiv.org/abs/math/9908150.

[41] Michael Meyer, Fabio Campos, and Steffen Reith. On lions and elligators: An efficient constant-time implementation of CSIDH. In Jintai Ding and Rainer Steinwandt, editors, *PQCrypto 2019*, pages 307–325, 2019. https://ia.cr/2018/1198.

[42] Michael Meyer and Steffen Reith. A faster way to the CSIDH. In *INDOCRYPT*, volume 11356 of *Lecture Notes in Computer Science*, pages 137–152. Springer, 2018. https://ia.cr/2018/782.

[43] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[44] Peter Lawrence Montgomery. *An FFT extension of the elliptic curve method of factorization*. PhD thesis, UCLA, 1992. https://cr.yp.to/bib/1992/montgomery.pdf.

[45] Dustin Moody and Daniel Shumow. Analogues of Vélu's formulas for isogenies on alternate models of elliptic curves. *Mathematics of Computation*, 85(300):1929–1951, 2016. https://ia.cr/2011/430.

[46] Jacques Morgenstern. Algorithmes linéaires tangents et complexité. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Série A*, 277:367–369, septembre 1973. https://gallica.bnf.fr/ark:/12148/cb34416987n/date.

[47] David B. Mumford. On the equations defining abelian varieties. I. *Inventiones Mathematicae*, 1(4):287–354, 1966. https://dash.harvard.edu/handle/1/3597241.

[48] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. (Short Paper) A faster constant-time algorithm of CSIDH keeping two points. In Nuttapong Attrapadung and Takeshi Yagi, editors, *Advances in Information and Computer Security*, pages 23–33, 2019. https://ia.cr/2019/353.

[49] John M. Pollard. Theorems on factorization and primality testing. *Mathematical Proceedings of the Cambridge Philosophical Society*, 76(3):521–528, 1974. https://doi.org/10.1017/S0305004100049252.

[50] Lawrence R. Rabiner, R. W. Schafer, and Charles M. Rader. The chirp-z transform algorithm. *IEEE Transactions on Audio and Electroacoustics*, 17:86–92, 1969. https://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/015_czt.pdf.

[51] Joost Renes. Computing isogenies between Montgomery curves using the action of (0, 0). PQCrypto 2018, pages 229–247, 2018. https://ia.cr/2017/1198.

[52] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies, 2006. https://ia.cr/2006/145.

[53] Volker Strassen. Einige Resultate über Berechnungskomplexität. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 78:1–8, 1976.

[54] Volker Strassen. The computational complexity of continued fractions. *SIAM Journal on Computing*, 12:1–27, 1983.

[55] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.0)*, 2020. https://www.sagemath.org.

[56] Jacques Vélu. Isogénies entre courbes elliptiques. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences, Série A*, 273:238–241, juillet 1971. https://gallica.bnf.fr/ark:/12148/cb34416987n/date.

Received 28 Feb 2020. Revised 28 Feb 2020.

DANIEL J. BERNSTEIN: *Department of Computer Science, University of Illinois at Chicago, USA*

and

Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
djb@cr.yp.to

LUCA DE FEO: *IBM Research Zürich, Switzerland*
antsXIV@defeo.lu

ANTONIN LEROUX: antonin.leroux@polytechnique.org
DGA, Inria and École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

BENJAMIN SMITH: smith@lix.polytechnique.fr
Inria and École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

VOLUME EDITORS

Stephen D. Galbraith
Mathematics Department
University of Auckland
New Zealand
https://orcid.org/0000-0001-7114-8377

The cover image is based on an illustration from the article "Supersingular curves with small noninteger endomorphisms", by Jonathan Love and Dan Boneh (see p. 9).

Electronic copies can be obtained free of charge from http://msp.org/obs/4 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: 978-1-935107-07-1 (print), 978-1-935107-08-8 (electronic)

First published 2020.

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 4

Fourteenth Algorithmic Number Theory Symposium

The Algorithmic Number Theory Symposium (ANTS), held biennially since 1994, is the premier international forum for research in computational and algorithmic 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 fourteenth ANTS meeting, which took place 29 June to 4 July 2020 via video conference, the plans for holding it at the University of Auckland, New Zealand, having been disrupted by the COVID-19 pandemic. The volume contains revised and edited versions of 24 refereed papers and one invited paper presented at the conference.

TABLE OF CONTENTS