Endomorphisms via splittings

Sabrina Kunzweiler[1,2], Min-Yi Shen[3]

[1] Centre Inria de l'université de Bordeaux
[2] Université de Bordeaux
[3](no affiliation)
sabrina.kunzweiler@math.u-bordeaux.fr
min.yi.shen0403106@gmail.com

Abstract. One of the fundamental hardness assumptions underlying isogeny-based cryptography is the problem of finding a non-trivial endomorphism of a given supersingular elliptic curve. We show that this problem is related to the problem of finding a *good* splitting of a principally polarized superspecial abelian surface. We provide formal security reductions, as well as a proof-of-concept implementation of an algorithm to compute endomorphisms of elliptic curves by solving the splitting problem.

Keywords Isogeny-based cryptography, abelian varieties, security assumptions, implementation

1 Introduction

Computing the endomorphism ring of a supersingular elliptic curve is one of the fundamental problems underlying the security of isogeny-based cryptography. The best known algorithms to solve this problem rely on finding enough non-trivial endomorphisms. Indeed, one can show that the so-called `OneEnd` problem which asks to find one non-trivial endomorphism is equivalent to the problem of computing the entire endomorphism ring [30].

Algorithmic solutions to `OneEnd` mostly rely on finding cycles in suitably chosen isogeny graphs. Given a supersingular elliptic curve $E/\mathbb{F}_{p^2}$, all of these algorithms have exponential runtime: In 1996, Kohel [21] first gave an algorithm with running time $O(p^{1+\epsilon})$, $\epsilon > 0$, for computing a non-trivial endomorphism of E. In 2016, Delfs and Galbraith [13] analyzed the structure of the isogeny graphs over $\mathbb{F}_p$ and gave a path finding algorithm in this graph which runs in $\tilde{O}(p^{1/4})$. This algorithm is then used as a subroutine in order to find an endomorphism for some elliptic curve over $\mathbb{F}_{p^2}$. First one computes two distinct isogeny paths to the $\mathbb{F}_p$-subgraph, and then one applies Delfs-Galbraith to obtain a path between the two $\mathbb{F}_p$-curves. The overall complexity is dominated by the first step, and it is given by $\tilde{O}(p^{1/2})$ under the generalized Riemann hypothesis (GRH). Further improvements of the algorithm (with the same asymptotic complexity) were provided by Corte-Real Santos, Castello and Shi in [9]. In 2020, Eisenträger, Hallgren, Leonardi, Morrison and Park [15] introduced a new cycle-finding algorithm that requires less memory, but still runs in time $\tilde{O}(p^{1/2})$. Recently, Fuselier, Iezzi, Kozek, Morrison and Namoijam [18] provided a new algorithm to compute an

inseparable endomorphism of a E. This algorithm is faster in practice than Delfs-Galbraith method and the cycle-finding algorithm because it only needs one path from the given curve over $\mathbb{F}_{p^2}$ to another curve over $\mathbb{F}_p$.

Another seemingly unrelated hardness assumption appears in isogeny-based protocols that work in the isogeny graphs of abelian surfaces. Given a (superspecial) irreducible principally polarized abelian surface A, it is presumably hard to find a splitting, i.e. an isogeny to a product of elliptic curves $\phi : A \to E_1 \times E_2$. We refer to this problem as the `Splitting` problem.

`Splitting` first appeared in the cryptanalysis of the 2-dimensional version of the Charles-Goren-Lauter (CGL) hash function [37],[7]. An important security assumption underlying this hash function is the hardness of finding an isogeny between superspecial principally polarized (p.p.) abelian surfaces, $\Phi : A \to A'$. In [10], Costello and Smith observed that this problem can be reduced to a problem in dimension one, by finding splittings $\phi : A \to E_1 \times E_2$ and $\phi' : A' \to E_1' \times E_2'$. The best known attack to solve `Splitting` relies on a random walk in the superspecial isogeny graph, and has complexity $O(p)$ when working over the finite field $\mathbb{F}_{p^2}$. Improvements on the concrete running time for this attack are provided by Corte-Real Santos, Costello and Frengley in [8].

Our contribution

In this work, we propose a new algorithm to find a non-trivial endormorphism of an elliptic curve. Our algorithm uses the connection between isogeny diamonds of elliptic curves with 2-dimensional product isogenies that was established by Kani [20]. To make this more precise, consider an elliptic curve E, a separable endomorphism $\phi \in \mathrm{End}(E)$, and choose a (potentially trivial) decomposition $\phi = \psi_2 \circ \phi_1$. These isogenies fit into a commutative diagram:

$$\begin{array}{ccc} E & \xrightarrow{\phi_1} & E_1 \\ \downarrow{\scriptstyle \phi_2} & & \downarrow{\scriptstyle \psi_2} \\ E_2 & \xrightarrow{\psi_1} & E \end{array}$$

Here $\deg(\phi_i) = \deg(\psi_i)$ for $i = 1, 2$. Such a configuration induces a product isogeny $\Phi : E \times E \to E_1 \times E_2$ defined as

$$\Phi : \begin{pmatrix} P \\ Q \end{pmatrix} \mapsto \begin{pmatrix} \phi_1 & \hat{\psi}_2 \\ -\phi_2 & \hat{\psi}_1 \end{pmatrix} \cdot \begin{pmatrix} P \\ Q \end{pmatrix}.$$

Instead of searching for cycles in the isogeny graph involving the elliptic curve E, we will search for product isogenies in the isogeny graph of dimension 2. A rough outline of our appraoch is provided in Algorithm 1 (FindEndomorphism). The idea is simple, first we compute a gluing $E \times E \to \mathrm{Jac}(C)$, and then search for a *good* splitting $\mathrm{Jac}(C) \to E_1 \times E_2$ for some elliptic curves E_1, E_2 (see Problem 4.5).

In order to be a solution to the `OneEnd` problem, it is required that the output of FindEndomorphism is a *non-trivial* endomorphism. While this might not be the case in general, we provide an explicit version of the algorithm using chains of $(2, 2)$-isogenies, where this subtlety is resolved naturally (Algorithm 6).

Algorithm 1 FindEndomorphism *(sketch of the general idea)*

Input: E a supersingular elliptic curve over a finite field $\mathbb{F}_{p^2}$.
Output: $\phi \in End(E)$.

1: Compute an (N_1, N_1)-gluing $f_1 : E \times E \to \mathrm{Jac}(C)$ for some genus-2 curve C, and some $N_1 \in \mathbb{N}$
2: Find a *good* (N_2, N_2)-splitting $f_2 : \mathrm{Jac}(C) \to E_1 \times E_2$ for some $N_2 \in \mathbb{N}$, i.e. it is required that

$$f_2 \circ f_1 : E \times E \to E_1 \times E_2$$

is an (N_1N_2, N_1N_2)-isogeny.
3: Use Kani's lemma to translate the product isogeny $f_2 \circ f_1$ to an endomorphism $\phi : E \to E$.

The dominating step in Algorithm 1 is Step 2 which requires to solve the `Splitting` problem. With the currently best known methods to solve this problem, our algorithm is not competitive with state-of-the-art methods to solve `OneEnd`. Instead, the main purpose of our paper is to provide new insight into the endomorphism problem. In particular, it proves a reduction of `OneEnd` to `Splitting`. The problem `OneEnd` is well-studied and also appears in classical algorithmic number theory, whereas the `Splitting` problem is less studied. Moreover `Splitting` is a very different problem in nature. It can be phrased as a more geometric problem which means that there are completely different tools that could be helpful to solve the problem. An example is given by the use of refined Humbert invariants (see for example [8], [23]).

We make the reduction of `OneEnd` to a variant of `GoodSplitting` explicit. While the focus of our paper is on the `OneEnd` problem, we also provide an analogous algorithm which can be used to prove a reduction from the isogeny-finding problem `Isogeny` to `Splitting`. Furthermore, we provide a proof-of-concept implementation in SageMath. This implementation can be used to find an efficient representation of a non-scalar endomorphism of a supersingular elliptic curve defined over $\mathbb{F}_{p^2}$ (provided that $p \cong 3 \pmod 4$). The code is publicly available in our GitHub repository

`https://github.com/easonmath/endomorphisms-via-splittings`

Using this implementation, we conducted experiments over different finite fields. We compare two different versions of the algorithm to find a *good* splitting in our setting: the faster method is based on a long random walk. The second method is slower but has asymptotically the same runtime. It searches splittings of bounded degree. The latter yields an efficient representation of the corresponding endomorphism. Interestingly, the degrees of the recovered endomorphisms seem to be smaller than the degrees obtained from other methods in the literature. Moreover there is no reason for the degrees to be smooth and our method even allows to find endomorphisms of prime degree.

Conclusion and future work

One of the main takeaways of this work should be that it is important to understand the splitting problem (in dimension 2) in order to understand the hardness of the endomorphism ring problem (in dimension 1). This claim is supported by a concrete

algorithm (including an implementation) to compute endomorphisms via splittings, as well as proven security reductions. We refer to Figure 4 for an overview.

While the resulting algorithm to compute endomorphisms is asymptotically slower than state-of-the-art algorithms, we believe that the general method could lead to new competitive algorithms. For instance, the following idea was pitched to us by Damien Robert: In Step 1 of Algorithm 1, one could replace $E \times E$ by $E \times E^\sigma$, where E^σ is the Frobenius twist of E. This p.p. abelian surface is defined over $\mathbb{F}_p$, hence one can search for a splitting in a much smaller isogeny graph. A concrete analysis of the running time and an implementation of this modification are a work in progress.

An open problem for future research is the study of the relation of `Splitting` with other hardness assumptions in dimension 2. We briefly elaborate on the relation with the endomorphism ring problem in dimension 2 (Remark 4.13). A more complete analysis and formal reductions are left for future work.

Related work

The first time Kani's lemma appeared in isogeny-based cryptography was in the attacks on the key exchange protocol SIDH in 2022 [5,25,32]. Since then, many constructions appeared that make use of the bridge between isogenies of dimension one and isogenies in higher dimensions, most prominently the signature scheme SQISignHD [11], and its 2-dimensional variants [3,14,29].

Regarding the connection between one-dimensional problems and two-dimensional problems, there is a recent preprint by Kırımlı and Martindale [23]. Under GRH, they show that finding an isogeny between two given supersingular elliptic curves E_1 and E_2 over $\mathbb{F}_{p^2}$ is equivalent to computing the coefficients of a *refined Humbert invariant* $q_{(E_1 \times E_2, \theta_{E_1} \times \theta_{E_2})}$ of the p.p. abelian surface $(E_1 \times E_2, \theta_{E_1} \times \theta_{E_2})$. The former, referred to as `Isogeny` in this paper, is a well-known hard problem in dimension one. The latter asks one to compute an isomorphism invariant, introduced by Kani [19], on the polarized Néron-Severi group of a given p.p. abelian surface.

Outline

In Section 2, we recall results from the literature related to isogenies of p.p. abelian surfaces. In Section 3, we provide an explicit description of Algorithm 1 that builds the basis of our implementation. In Section 4, we prove reductions between isogeny problems in dimension 1 and the splitting problem in dimension 2. In Section 5, we discuss data collected in our experiments for finding endomorphisms. Furthermore, the appendix contains more details on our implementation.

Acknowledgments

Sabrina Kunzweiler received funding from the French National Research Agency (ANR) under the ANR CIAO with reference ANR-19-CE48-0008, and the France 2030 program under grant ANR-22-PETQ-0008 (PQ-TLS).

Disclaimer Part of this article is published in the second author's Master's thesis [35].

2 Preliminaries

In this section, we introduce terminology and recall some results from the literature related to isogenies of (superspecial) principally polarized abelian surface. In particular, we recall Kani's lemma, and a method to compute 2-isogenies using theta coordinates which forms the basis of our implementation. A reader only interested in the formal reductions in Section 4 may skip the part about theta coordinates (and Section 3).

2.1 Isogenies of principally polarized abelian surfaces

Let A be a principally polarized abelian surface. An isogeny $f : A \to B$ is defined by its kernel $K = \ker(f)$ up to isomorphism. The latter is a maximal N-isotropic subgroup of A for some $N \in \mathbb{N}$. We say that f is an (N, N)-*isogeny* if the kernel K has rank 2, equivalently $\ker(f) \cong \mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$. Similarly, an isogeny of elliptic curves $f : E \to E'$ is called an N-*isogeny* if the kernel is a cyclic group of order N.

Consider an (N_1, N_1)-isogeny $f_1 : A \to B$ and an (N_2, N_2)-isogeny $f_2 : B \to C$. We say that f_2 is a *good extension* of f_1 if $f_2 \circ f_1$ is an (N_1N_2, N_1N_2)-isogeny. Note that if $\gcd(N_2, N_1) = 1$, then any f_2 is necessarily a good extension of f_1 (in the above setting). This is not true if $\gcd(N_1, N_2) > 1$, see for example [4, Section 2.2].

2.2 Kani's lemma

In [20], Kani studied genus-2 curves that admit an elliptic subcover. Among other results, the paper contains an equivalence relation between certain product isogenies in dimension 2 and isogeny-diamond configurations.

Definition 2.1. *Let $d_1, d_2 \in \mathbb{N}$ be two positive integers. A* (d_1, d_2)-isogeny diamond *is given by the decomposition of a $d_1 \cdot d_2$-isogeny $\phi : E \to E'$ as*

$$\phi = \psi_2 \circ \phi_1 = \psi_1 \circ \phi_2,$$

with

$$\deg(\phi_1) = \deg(\psi_1) = d_1, \quad \deg(\phi_2) = \deg(\psi_2) = d_2.$$

We represent (d_1, d_2)-isogeny diamonds by a commutative diagram:

$$\begin{array}{ccc} E & \xrightarrow{\phi_1} & E_1 \\ \downarrow{\scriptstyle \phi_2} & & \downarrow{\scriptstyle \psi_2} \\ E_2 & \xrightarrow{\psi_1} & E' \end{array}$$

Further, we say that the above configuration is an isogeny diamond configuration of order $N = d_1 + d_2$ from E to E'.

We remark that in [20], there is an additional condition that $\ker(\phi_1) \cap \ker(\phi_2) = \{0\}$. This leads to a correspondence with more specific product isogenies [20, Corollary 2.4]. Instead, here we use the more general setting.

Lemma 2.2. *Given two elliptic curves E, E', there is a 1-to-1 correspondence between equivalence classes of isogeny diamond configurations of order N from E to E', and maximal N-isotropic subgroups $G \subset (E \times E')[N]$ defining a non-diagonal product isogeny with domain $E \times E'$.*

Explicitly, given a (d_1, d_2)-isogeny diamond with notation as in Definition 2.1, the induced product isogeny is given by

$$F : E \times E' \longrightarrow E_1 \times E_2 \quad \text{with } F = \begin{pmatrix} \phi_1 & \tilde{\psi}_2 \\ -\phi_2 & \tilde{\psi}_1 \end{pmatrix}.$$

Proof. This follows from Theorem 2.3 in [20]. □

2.3 Level-2 theta coordinates

For the instantiation of our algorithm, we use $(2, 2)$-isogenies. Such computations can be performed efficiently using level-2 theta coordinates. Here, we informally introduce the theory of level-2 theta coordinates, and briefly summarize the necessary algorithms from the literature.

Theta functions provide a convenient way to work with principally polarized (p.p.) abelian varieties in any dimension. In this manuscript, we denote by θ^A a *level*-2 theta structure attached to the abelian variety A of dimension g. This defines an embedding of the Kummer variety, $A/\langle \pm 1 \rangle$, into the projective space $\mathbb{P}^{2^g-1}$. This means that a point of an elliptic curve E is represented by two projective coordinates, $\theta^E(P) = (x_0 : x_1)$, and a point on an abelian surface A is represented by four projective coordinates $\theta^A(P) = (x_0 : x_1 : x_2 : x_3)$.[1] We refer to these as (level-2) theta coordinates.

Theta null point We say that the theta coordinates of the zero element, $\theta^A(0_A)$, is the (level-2) *theta null point* of A. We denote $\theta^A(0_A) = (a_0 : a_1)$ for elliptic curves, and $\theta^A(0_A) = (a_0 : a_1 : a_2 : a_3)$ for p.p. abelian surfaces. Remarkably, the theta null point $\theta^A(0_A)$ determines the underlying p.p. abelian variety A up to isomorphism if A is irreducible [17,39] or $g \leq 3$ [26]. This is the case in all of our applications.

Product structure For a reducible abelian variety $A = A_1 \times A_2$, the theta structures on A_1 and A_2 induce a theta structure on A via the Segre embedding. A theta structure obtained in this way is called *product theta structure*. More explicitly, consider two elliptic curves equipped with a level-2 theta structure (E, θ^E) and $(E', \theta^{E'})$, and their product $A = E \times E'$. We write $\theta^A = \theta^E \times \theta^{E'}$ for the product theta structure. For a point $(P, P') \in A$, we have

$$\theta^A((P, P')) = (x_0 x'_0 : x_0 x'_1 : x_1 x'_0 : x_1 x'_1),$$

where $\theta^E(P) = (x_0 : x_1)$, $\theta^{E'}(P') = (x'_0 : x'_1)$.

[1] Of course, this representation is only well defined up to sign.

Symplectic transformations Different level-2 theta structures for the same abelian variety are related by symplectic transformations. Roughly, such transformations consist of scaling the coordinates by a 4th root of unity, applying a Hadamard transformation, or permuting the coordinates. For details on such transformations, we refer to [31, Appendix B] for a general description, and [22, Appendix A.1] for a description specific to theta structures of level 2. In our applications, this will become important when computing non-generic isogenies. In these cases, we need to either compute a non-product structure (gluings) or recover the product structure (splittings).

2.4 Radical isogeny formulas

Here, we summarize the radical isogeny formulas from [22], see also [31]. In any dimension g, a radical 2-isogeny $f : A \to B$ can be computed as the composition of three simple operations: coordinate-wise squaring, a Hadamard transform and taking coordinate-wise square-roots. The in- and output are theta null points $a = \theta^A(0_A)$ and $b = \theta^B(0_B)$, respectively. This is sketched in Figure 1.

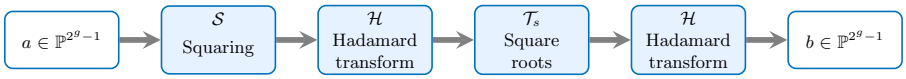

Fig. 1. Radical 2-isogeny $f : A \to B$ with $a = \theta^A(0_A)$ and $b = \theta^B(0_B)$ as described in [22, p.2]

In full generality, the definition of the square-root function, $\mathcal{T}_s$, is not straightforward, since certain square-roots have to be chosen compatibly. In this article, we only work with radical isogenies in dimension $g = 2$, where these subtleties do not arise yet. Thus, the description for these isogenies is particularly simple. It is outlined in Algorithm 2.

Note that the case distinction in Line 3 of Algorithm 2 is necessary to include non-generic isogenies. More precisely, the coordinates $x_0, \ldots, x_3$ computed in Line 2 are all non-zero if the input A is irreducible, and at most one coordinate may vanish if A is reducible.

Remark 2.3. The kernel of the isogeny $f : A \to B$ computed in Algorithm 2 is given by

$$\ker(f) = \langle (a_0 : -a_1 : a_2 : -a_3), (a_0 : a_1 : -a_2 : -a_3) \rangle.$$

This means that the isomorphism class of the codomain B does not depend on the choice of signs s_1, s_2, s_3. However, the choice of signs does determine the level-2 theta structure induced on B. In particular, it determines the kernel of the next $(2, 2)$-isogeny computation if a chain of $(2, 2)$-isogenies is computed, see [22, Section 3.5].

Algorithm 2 Radical2Isogeny (see [22, Algorithm 2] and [22, Appendix B])

Input: A level-2 theta null point $\theta^A(0_A) = (a_0 : a_1 : a_2 : a_3)$ of an abelian surface A and choice bits $s_i \in \{-1, 1\}^3$

Output: A level-2 theta null point $\theta^B(0_B) = (b_0 : b_1 : b_2 : b_3)$

1: $a'_0, a'_1, a'_2, a'_3 \leftarrow a_0^2, a_1^2, a_2^2, a_3^2$ ▷ Squaring

2: $x_0, x_1, x_2, x_3 \leftarrow a'_0 + a'_1 + a'_2 + a'_3,\ a'_0 - a'_1 + a'_2 - a'_3,\ a'_0 + a'_1 - a'_2 - a'_3,\ a'_0 - a'_1 - a'_2 + a'_3$ ▷ Hadamard

3: **if** $x_0 \neq 0$ **then**

4: $\quad y_0, y_1, y_2, y_3 \leftarrow x_0, s_1 \cdot \sqrt{x_0 x_1}, s_2 \cdot \sqrt{x_0 x_2}, s_3 \cdot \sqrt{x_0 x_3}$ ▷ Square-roots

5: **else**

6: $\quad y_0, y_1, y_2, y_3 \leftarrow 0, x_1, s_2 \cdot \sqrt{x_1 x_2}, s_3 \cdot \sqrt{x_1 x_3}$

7: **end if**

8: $b_0, b_1, b_2, b_3 \leftarrow y_0 + y_1 + y_2 + y_3,\ y_0 - y_1 + y_2 - y_3,\ y_0 + y_1 - y_2 - y_3,\ y_0 - y_1 - y_2 + y_3$ ▷ Hadamard

9: **return** $(b_0 : b_1 : b_2 : b_3)$

Remark 2.4. Let (A, θ^A) be a reducible p.p. abelian surface $A = E \times E'$ equipped with the product theta structure $\theta^A = \theta^E \times \theta^{E'}$, and consider the isogeny $f : A \to B$ computed as described in Algorithm 2.

From Remark 2.3, we know the kernel of this isogeny. Writing $\theta^E(0) = (e_0 : e_1)$ and $\theta^{E'}(0) = (e'_0 : e'_1)$, we obtain

$$\begin{aligned}\ker(f) =& \langle (e_0 e'_0 : -e_0 e'_1 : e_1 e'_0 : -e_1 e'_1),\ (e_0 e'_0 : e_0 e'_1 : -e_1 e'_0 : -e_1 e'_1) \rangle \\ =& \langle ((e_0 : e_1), (e'_0 : -e'_1)),\ ((e_0 : -e_1), (e'_0 : e'_1)) \rangle.\end{aligned}$$

This shows that f is a diagonal isogeny and the codomain is again a product of elliptic curves:

$$f : E \times E' \to E/\langle (e_0 : -e_1) \rangle \times E'/\langle (e'_0 : e'_1) \rangle.$$

Note that the induced theta structure on the codomain is not necessarily a *product* theta structure, hence iteratively applying the radical isogeny formula still leads to irreducible p.p. abelian surfaces. In our applications, it will be preferable to compute a radical gluing immediately. This can be achieved by applying a symplectic automorphism in order to change the underlying theta structure. We make this explicit in Algorithm 3 (Gluing).

2.5 Evaluation of isogenies

For evaluating a 2-isogeny which was computed by a radical formula, we apply the algorithms developed in [12], see also [31]. Given a radical 2-isogeny $f : A \to B$ computed as in Figure 1 (see also Algorithm 2), then (in the generic case) the evaluation of this isogeny at a point can be expressed in a similar way. Only the coordinate-wise square-roots are replaced by coordinate-wise scaling by the dual theta null point of the codomain, $\tilde{b}$. We recall that $\tilde{b}$ is obtained from the null point $b = \theta^B(0_B)$ by applying a Hadamard transformation. The evaluation of the 2-isogeny f is sketched in Figure 2.

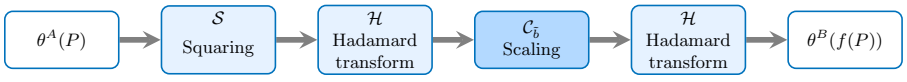

Fig. 2. Evaluation of a (generic) 2-isogeny $f : A \to B$ at a point $P \in A$, following [31, §5]

Remark 2.5. The description in Figure 2 applies to all 2-isogenies $f : A \to B$ where A is irreducible. On the other hand, when A is the product of elliptic curves, then one of the coordinates of the dual theta null point, $\tilde{b}$, might vanish. In this case, the scaling step $\mathcal{C}_{\tilde{b}}$ is not well-defined, and the evaluation of isogenies requires more care. One possibility is to work with additional information on a 4-torsion point that lies above the kernel of the isogeny f. This idea, which we use in our implementation (Algorithm 14), is explained in [12, pp. 17-18].

In our applications, it is also required to evaluate the dual isogeny of some isogeny $f : A \to B$ (computed as in Figure 1). Similarly to the evaluation of f, it can be decomposed into 4 basic operations. This is depicted in Figure 3. The description follows immediately from the duplication formulas [31].

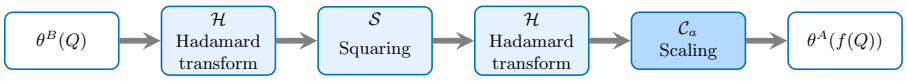

Fig. 3. Evaluation of the dual isogeny $\hat{f} : B \to A$ at a point $Q \in B$, where $f : A \to B$ is as in Figure 2.

We remark that this description is only valid when none of the coordinates of the theta null point $a = \theta^A(0)$ vanishes. Here, the case that needs to be treated separately is when B is reducible (i.e. $\hat{f}$ is a gluing isogeny).

3 A concrete algorithm: endomorphisms via splittings

Our general idea to find an endomorphism of an elliptic curve is sketched in Algorithm 1 in the introduction. First, we compute an (N_1, N_1)-gluing $E \times E \to \mathrm{Jac}(C)$ for some $N_1 \in \mathbb{N}$. And then we find a (good) (N_2, N_2)-splitting $\mathrm{Jac}(C) \to E_1 \times E_2$. Composition yields an (N_1N_2, N_1N_2)-isogeny $E \times E \to E_1 \times E_2$ which is necessarily induced by an isogeny diamond:

$$\begin{array}{ccc} E & \xrightarrow{\phi_1} & E_1 \\ \downarrow{\scriptstyle \phi_2} & & \downarrow{\scriptstyle \psi_2} \\ E_2 & \xrightarrow{\psi_1} & E \end{array}$$

From this isogeny diamond we can read off the endomorphism $\psi_2 \circ \phi_1 \in \mathrm{End}(E)$.

3.1 Instantiation of Algorithm 1

There are several choices that can be made in the implementation of Algorithm 1. In the following, we describe a solution based on the computation of $(2,2)$-isogenies in level-2 theta coordinates. The computed isogeny chain will be of the form

$$E \times E \xrightarrow{\sigma} (E \times E)_\sigma \xrightarrow{f_1} A_1 \to \ldots \to A_{n-1} \xrightarrow{f_n} (E_1 \times E_2)_{\tau^{-1}} \xrightarrow{\tau} E_1 \times E_2, \quad (1)$$

where σ and τ are automorphisms that change the underlying theta structure, $f_1, \ldots, f_n$ are $(2,2)$-isogenies, and $A_1 \ldots A_{n-1}$ are irreducible p.p. abelian surfaces.

Remark 3.1. We say the isogeny chain (Eq. 1) is of length n, since σ and τ are isomorphisms. We also fix the notation of σ and τ in this paper; note that σ is a fixed automorphism discussed below while τ depends on the surface $(E_1 \times E_2)_{\tau^{-1}}$.

Gluing The first steps, applying the autmorphism σ and computing the isogeny f_1, are summarized in Algorithm 3.

Algorithm 3 Gluing

Input: Theta null points of two elliptic curves $\theta^E(0) = (e_0 : e_1)$ and $\theta^{E'}(0) = (e'_0 : e'_1)$, with $j(E) \neq 0, 1728$ or $j(E') \neq 0, 1728$.
Output: Theta null point of an irreducible p.p. abelian surface $\theta(B) = (b_0 : b_1 : b_2 : b_3)$ that is 2-isogenous to $E \times E'$.

1: $\mathsf{NL}'_0 = (a'_0 : a'_1 : a'_2 : a'_3) \leftarrow (e_0e'_0 : e_0e'_1 : e_1e'_0 : e_1e'_1)$ ▷ Segre embedding
2: $\mathsf{NL}_0 = (a_0 : a_1 : a_2 : a_3) \leftarrow (a'_0 + a'_1 + a'_2 - a'_3 : -i(a'_0 - a'_1 + a'_2 + a'_3) : a'_0 - a'_1 - a'_2 - a'_3 : -i(a'_0 + a'_1 - a'_2 + a'_3))$ ▷ symplectic automorphism
3: $\mathsf{NL}_1 = (b_0 : b_1 : b_2 : b_3) \leftarrow \mathsf{Radical2Isogeny}(a_0 : a_1 : a_2 : a_3)$
4: **return** $\mathsf{NL}'_0, \mathsf{NL}_0, \mathsf{NL}_1$

Lemma 3.2. *On input theta null points of two elliptic curves $\theta^E(0) = (e_0 : e_1)$ and $\theta^{E'}(0) = (e'_0 : e'_1)$, Algorithm 3 (Gluing) outputs:*

- NL_0, *theta null point of $E \times E'$ equipped with the product theta structure,*
- NL'_0, *theta null point equivalent to NL_0 obtained via a symplectic transformation,*
- NL_1, *theta null point of a p.p. abelian surface B that is 2-isogenous to $E \times E'$.*

Moreover, the kernel of the $(2,2)$-isogeny $f : E \times E' \to B$ is given by

$$\ker(f) = \langle ((e_1 : e_0), (-e'_0 : e'_1)), ((-e_0 : e_1), (e'_1 : e'_0)) \rangle.$$

The codomain B is irreducible if one of the following conditions holds:

1. $E \not\cong E'$.
2. $j(E) = j(E') \notin \{0, 1728\}$, *and* $(e_0 : e_1) = (e'_0 : e'_1)$.

Proof. First, we note that the computation in Line 2 corresponds to applying a symplectic automorphism σ with matrix representation

$$M(\sigma) = \begin{pmatrix} 1 & 1 & 1 & -1 \\ -i & i & -i & -i \\ 1 & -1 & -1 & -1 \\ -i & -i & i & -i \end{pmatrix}.$$

One can verify that

$$\sigma(\iota((e_1 : e_0), (-e_0' : e_1'))) = (-a_0 : -a_1 : a_2 : a_3),$$
$$\sigma(\iota((-e_0 : e_1), (e_1' : e_0'))) = (-a_0 : a_1 : a_2 : -a_3),$$

where ι denotes the Segre embedding. These elements generate the kernel of the radical isogeny applied in Line 3 (see Remark 2.3).

It remains to show that the codomain of the isogeny f is irreducible. As we have just shown, the kernel of f is non-diagonal. If $E \not\cong E'$, this readily implies that f is a gluing. On the other hand, when $E \cong E'$, there exists an isogeny diamond of degree-1 isogenies which induces a product isogeny $\tilde{f}$. As long as $j(E) = j(E') \notin \{0, 1728\}$, this product isogeny is unique with kernel

$$\ker(\tilde{f}) = \langle((-e_0 : e_1), (-e_0 : e_1)), ((e_1 : e_0), (e_1 : e_0))\rangle,$$

assuming $(e_0' : e_1') = (e_0 : e_1)$ (see also [12, Appendix A.3]). In particular $\ker(\tilde{f}) \neq \ker(f)$. □

For simplicity, the case $j(E) = j(E') \in \{0, 1728\}$ is excluded in Lemma 3.2, since it does not appear in our applications (we already know non-trivial endomorphisms of these elliptic curves). The reader is referred to the description of $(2, 2)$-isogenies with domain $E \times E$ in Weierstrass coordinates provided in [16, Tables 9 and 10].

Finding a good splitting The next step is the bottleneck of our method. Given a superspecial irreducible p.p. abelian surface, we want to find a splitting. Asymptotically, the best known method to do this is by taking random walks in the superspecial isogeny graph [10],[8]. A basic method is outlined in Algorithm 4 (FindSplitting). This algorithm is expected to return an isogeny after $O(p)$ steps. The reasoning behind this is that the proportion of reducible p.p. abelian surfaces in the superspecial isogeny graph over $\mathbb{F}_{p^2}$ is $O(1/p)$.

A drawback of Algorithm 4 is that the output is long and hence the evaluation of the corresponding endomorphism inefficient. In order to obtain a splitting that can be evaluated efficiently, we propose Algorithm 5 (FindSmallSplitting) which takes walks of bounded length N until a splitting is found. More precisely, it searches a splitting in a subtree of the superspecial isogeny graph with root $\mathsf{NL}_1 = \theta^{A_1}(0)$. The algorithm fails if none of the $O(8^N)$ inspected p.p. abelian surfaces in the neighborhood of the starting vertex is reducible.

Algorithm 4 FindSplitting

Input: The theta null point $\mathsf{NL}_1 = \theta^{A_1}(0)$ of an irreducible superspecial p.p. abelian surface encoding an incoming $(2,2)$-isogeny $f_1 : A_0 \to A_1$

Output: A $(2^{n-1}, 2^{n-1})$-isogeny $g : A_1 \to E_1 \times E_2$ for some $n \in \mathbb{N}$ and elliptic curves E_1, E_2, so that $g \circ f_1$ is a $(2^n, 2^n)$-isogeny. The isogeny chain is described by a chain of theta null points $\mathsf{NL}_1, \ldots, \mathsf{NL}_n$.

$i \leftarrow 1$
while True **do**
  $S_i \xleftarrow{\$} \{-1, 1\}^3$
  $\mathsf{NL}_{i+1} \leftarrow \mathsf{Radical2Isogeny}(\mathsf{NL}_i, S_i)$
  if $\mathsf{IsReducible}(\mathsf{NL}_{i+1})$ **then**
    return $\mathsf{NL}_1, \ldots, \mathsf{NL}_{i+1}$
  end if
  $i \leftarrow i + 1$
end while

Algorithm 5 FindSmallSplitting

Input: The theta null point $\mathsf{NL}_1 = \theta^{A_1}(0)$ of an irreducible superspecial p.p. abelian surface encoding an incoming $(2,2)$-isogeny $f_1 : A_0 \to A_1$, and a bound $N \in \mathbb{N}$.

Output: A $(2^{n-1}, 2^{n-1})$-isogeny $g : A_1 \to E_1 \times E_2$ for some $1 \leq n \leq N$ and elliptic curves E_1, E_2, so that $g \circ f_1$ is a $(2^n, 2^n)$-isogeny. The isogeny chain is described by a chain of theta null points $\mathsf{NL}_1, \ldots, \mathsf{NL}_n$. If no such isogeny exists[2]the algorithm returns $\perp$.

$S = S_0, \ldots, S_7 \leftarrow \{-1, 1\}^3$ ▷ list of sign choices
$i \leftarrow 1$ ▷ current depth
$j_1 \leftarrow 0$ ▷ sign choice at step i
while $i > 0$ **do**
  if $j_i = 8$ or $i = N - 1$ **then**
    $j_{i-1} \leftarrow j_{i-1} + 1$
    $i \leftarrow i - 1$
  else
    $\mathsf{NL}_{i+1} \leftarrow \mathsf{Radical2Isogeny}(\mathsf{NL}_i, S_{j_i})$
    if $\mathsf{IsReducible}(\mathsf{NL}_{i+1})$ **then**
      return $\mathsf{NL}_1, \ldots, \mathsf{NL}_{i+1}$
    end if
    $j_{i+1} \leftarrow 0$
    $i \leftarrow i + 1$
  end if
end while
return $\perp$

Lemma 3.3. *If Algorithms 4 or 5 terminate successfully, then the output is correct.*

Proof. The only non-trivial part is to make sure that g is a $(2^{n-1}, 2^{n-1})$-isogeny, and the composition $g \circ f_1$ a $(2^n, 2^n)$-isogeny. This is an immediate consequence of [22, Section 3.4], where it is shown that the composition of the radical $(2,2)$-isogeny formulas

[2] Note that we do not test all possible $(2^n, 2^n)$-isogenies, see Remark 2.3.

underlying Algorithm 2 (Radical2Isogeny) yields a $(2^k, 2^k)$-isogeny. Note that the gluing $E \times E \to A$ is computed using the same formulas (after applying the isomorphism σ). □

Explicit endomorphism finding algorithm Our explicit version of the endomorphism finding algorithm is just a combination of the explicit gluing algorithm (Algorithm 3) and one of the splitting algorithms (Algorithms 4 and 5). For future reference, we state this explicitly in Algorithm 6. The output of this algorithm provides a list of theta null points NL. This output provides a representation of the endomorphism which is an *efficient representation* if the length of the chain is polynomial in $\log p$ (see also Definition 4.1).

Algorithm 6 FindEndomorphismExplicit

Input: E a supersingular elliptic curve over a finite field $\mathbb{F}_{p^2}$ with level-2 theta null point $(e_0 : e_1)$ and $j(E) \neq 0, 1728$

Output: A list of theta null points of abelian surfaces that represent a *non-trivial* endomorphism of E via Algorithm 14.

1: $\mathsf{NL}'_0, \mathsf{NL}_0, \mathsf{NL}_1 \leftarrow \mathsf{Gluing}((e_0 : e_1), (e_0 : e_1))$ ▷ Algorithm 3

2: $\mathsf{NL}_1, \ldots, \mathsf{NL}_n \leftarrow \mathsf{FindSplitting}(\mathsf{NL}_1)$ or $\mathsf{FindSmallSplitting}(\mathsf{NL}_1)$ ▷ Algorithm 4 or Algorithm 5

3: **return** $\mathsf{NL}'_0, \mathsf{NL}_0, \ldots, \mathsf{NL}_n$

Extracting the endomorphism from a product isogeny Assume we are given a chain of theta null points outputted by Algorithm 6. This chain induces a product isogeny $F \colon E \times E \to E_1 \times E_2$ with matrix representation

$$F = \begin{pmatrix} \phi_1 & \hat{\psi}_2 \\ -\phi_2 & \hat{\psi}_1 \end{pmatrix}.$$

By [33, Lemma 3], the adjoint isogeny $\tilde{F}$ has the matrix form $\begin{pmatrix} \hat{\phi}_1 & -\hat{\phi}_2 \\ \psi_2 & \psi_1 \end{pmatrix}$. With the knowledge of F and $\tilde{F}$, we are able to 'extract' the endomorphism $\psi_2 \circ \phi_1$. By 'extract' an endomorphism, we mean being able to evaluate the map on a given point. Briefly, we first compute $\phi_1(P)$ from the first component of $F \cdot \left(\begin{smallmatrix} P \\ 0_E \end{smallmatrix}\right)$ and then obtain the desired $\psi_2 \circ \phi_1(P)$ from

$$\tilde{F} \cdot \begin{pmatrix} \phi_1(P) \\ 0_{E_2} \end{pmatrix} = \begin{pmatrix} [\deg \phi_1](P) \\ -\psi_2 \circ \phi_1(P) \end{pmatrix}. \tag{2}$$

Recall that F was computed as a chain of $(2,2)$-isogenies (and two additional isomorphisms), more precisely $F = \tau \circ f_n \circ \cdots \circ f_1 \circ \sigma$ as in Equation 1. The evaluation of the individual $(2,2)$-isogenies is efficiently computed as $\mathcal{H} \circ \mathcal{C}_{\tilde{b}} \circ \mathcal{H} \circ \mathcal{S}$, where b is the theta null point of the corresponding codomain (see Figure 2). Only in the first step,

i.e. for the gluing isogeny f_1, this approach does not work (Remark 2.5). We explain the necessary modifications in detail in Appendix A.1. In particular, we describe the algorithm ChainEvaluation (Algorithm 13). The latter takes as input a chain of theta null points NL describing F as in (Eq. 1), and a point $(P, Q) \in E \times E$, and outputs the image $F((P, Q)) \in E_1 \times E_2$.

Similarly, the adjoint isogeny $\tilde{F} : E_1 \times E_2 \to E \times E$ is given by

$$\tilde{F} = \sigma^{-1} \circ \hat{f}_1 \circ \cdots \circ \hat{f}_n \circ \tau^{-1}.$$

The evaluation of the individual steps is efficiently computed as $\mathcal{C}_a \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H}$ as in Figure 3. Similar to before, only the first isogeny, here $\hat{f}_n$, requires special treatment since it represents a gluing isogeny. The details for this step are provided in Appendix A.2.

3.2 Constructing *non-scalar* endomorphisms

Assume we found an endomorphism $\psi_2 \circ \phi_1 \in \mathrm{End}(E)$ using Algorithm 6. In order for this to be a solution to the `OneEnd` problem, we require that the endomorphism is non-scalar. In this subsection, we prove that scalar endomorphisms are naturally avoided in our setup.

Lemma 3.4. *Let E be an elliptic curve and $F : E \times E \to E_1 \times E_2$ a $(2^n, 2^n)$-product isogeny. If $\psi_2 \circ \phi_1$ is scalar, then all $\phi_1, \phi_2, \psi_1, \psi_2$ are cyclic isogenies of degree 2^{n-1}.*

Proof. We denote $\psi_2 \circ \phi_1 = [N]$, and use the decompositions

$$\phi_1 = [a] \circ \phi_1', \quad \psi_2 = [b] \circ \psi_2',$$

where $[a], [b]$ denote scalar multiplications (by positive integers a, b) and ϕ_1', ψ_2' are cyclic isogenies. Note that

$$[N] = \psi_2 \circ \phi_1 = [ab] \circ \psi_2' \circ \phi_1'$$

which implies that $\hat{\psi_2'} = \pm\phi_1'$. In particular $\deg(\psi_2') = \deg(\phi_1')$. This provides us with a condition on the degrees:

$$2^n = \deg(\phi_1) + \deg(\psi_2) = \deg(\phi_1') \cdot (a^2 + b^2).$$

It follows that $\deg(\phi_1') = 2^{k_1}$ for some $k_1 < n$, and $a^2 + b^2 = 2^{n-k_1}$. The only solution to the latter is $a = b = 2^{k_1'}$ with $k_1' = (n - k_1 - 1)/2$.[3] Similarly, we write

$$\phi_2 = [c] \circ \phi_2', \quad \psi_1 = [d] \circ \psi_1',$$

where $[c], [d]$ denote scalar multiplications and ϕ_2', ψ_1' are cyclic isogenies, and obtain the condition $\deg(\phi_2') = 2^{k_2}$ for some $k_2 < n$ and $c = d = 2^{k_2'}$ with $k_2' = (n - k_2 - 1)/2$.

[3] One way to see this is writing $a = 2^\alpha(1 + 2a')$, $b = 2^\beta(1 + 2b')$ for some a', b', and w.l.o.g. $\beta \geq \alpha$. Then $a^2 + b^2 = 2^{2\alpha}(1 + 4(a' + a'^2) + 2^{2(\beta-\alpha)}(1 + 4(b' + b'^2)))$. So $a^2 + b^2$ can only be a power of two if $a' = b' = 0$ and $\alpha = \beta$.

In conclusion, the product isogeny is of the form

$$F = \begin{pmatrix} [2^{k_1'}]\phi_1' & \pm[2^{k_1'}]\phi_1' \\ [2^{k_2'}]\phi_2' & \pm[2^{k_2'}]\phi_2' \end{pmatrix}.$$

By assumption, F is a $(2^n, 2^n)$-isogeny. In particular, $\ker(F)[2] \subset (E \times E)[2]$ is maximal 2-isotropic. Now if $k_1' > 0$ or $k_2' > 0$, then the rank of $\ker(F)[2]$ is greater than 2. This proves that $k_1' = k_2' = 0$, hence $k_1 = k_2 = n - 1$. □

Lemma 3.5. *Let $f : E \times E \to A$ be a $(2,2)$-isogeny with kernel*

$$\ker(f) = \{(P, P) \mid P \in E[2]\},$$

then $A \cong E \times E$.

Proof. Consider the Kani diagram

$$\begin{array}{ccc} E & \xrightarrow{\cong} & E \\ \downarrow{\scriptstyle \cong} & & \downarrow{\scriptstyle \cong} \\ E & \xrightarrow{\cong} & E \end{array}.$$

This induces the $(2,2)$-isogeny

$$F : E \times E \to E \times E \quad \text{with } F = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}.$$

Note that

$$F(P, P) = (P + P, -P + P) = ([2]P, 0) \quad \text{for all } P \in E.$$

Consequently,

$$\ker(F) = \{(P, P) \mid P \in E[2]\},$$

and the isogenies F and f coincide (up to isomorphism). □

Using the preceding lemmas, we can now show that Algorithm 6 finds a non-scalar endomorphism.

Proposition 3.6. *Let $\psi_2 \circ \phi_1 \in \mathrm{End}(E)$ be an endomorphism outputted by Algorithm 6. Then $\psi_2 \circ \phi_1$ is non-scalar.*

Proof. Let $F : E \times E \to E_1 \times E_2$ be a $(2^n, 2^n)$-product isogeny constructed in Algorithm 6, and let $F = f_n \circ \cdots \circ f_1$, i.e.

$$E \times E \xrightarrow{f_1} A_1 \xrightarrow{f_2} \ldots \xrightarrow{f_{n-1}} A_{n-1} \xrightarrow{f_n} E_1 \times E_2$$

be the decomposition into $(2,2)$-isogenies. Assume that $\psi_2 \circ \phi_1$ is a scalar endomorphism. Note that in our algorithm, the isogeny f_1 is a gluing, i.e. A_1 is irreducible.

On the other hand, it follows from Lemma 3.4 that $F = \begin{pmatrix} \phi_1 \pm \phi_1 \\ -\phi_2 \pm \phi_2 \end{pmatrix}$ for some 2^{n-1}-isogenies ϕ_1, ϕ_2. For a 2-torsion point $P \in E[2]$, it holds that

$$F((P,P)) = \begin{pmatrix} \phi_1(P) \pm \phi_1(P) \\ -\phi_2(P) \pm \phi_2(P) \end{pmatrix} = \begin{pmatrix} \phi_1(P \pm P) \\ \phi_2(-P \pm P) \end{pmatrix} = \begin{pmatrix} \phi_1(0) \\ \phi_2(0) \end{pmatrix} = 0$$

Now Lemma 3.5 implies that the codomain of f_1 is $A_1 = E \times E$. This contradicts the fact that f_1 is a gluing, hence the endomorphism $\psi_2 \circ \phi_1$ is non-scalar.

□

3.3 Analyzing the degree of the endomorphism

The degree of the endomorphism that we find using Algorithm 6, depends on the degree of the $(2^n, 2^n)$-product isogeny. More precisely, we find an endomorphism $\psi_2 \circ \phi_1$ of degree $d = \deg(\psi_2) \cdot \deg(\phi_1)$, where $\deg(\psi_2) + \deg(\phi_1) = 2^n$. This provides us with the bounds

$$2^n < d < 2^{2n}.$$

Order of magnitude We know that the proportion of products of supersingular curves in the superspecial isogeny graph [10, Section 6] is

$$\frac{\#S_2(p)^E}{\#S_2(p)} \approx \frac{10}{p},$$

We expect to find a splitting after inspecting $O(p)$ vertices in the graph. Using Algorithm 4 to find a (long) splitting, we expect to succeed for a $(2^n, 2^n)$-isogeny with $n \approx p$. The degree of the resulting endomorphism is exponential in p.

On the other hand, using Algorithm 5 to find a (short) $(2^n, 2^n)$-splitting bounded by $n \leq N$, we visit $O(8^N)$ vertices in the neighborhood of the starting vertex. Under the (strong) assumption that these vertices behave sufficiently random, we would expect that a value of $N = \log_2(p)/3$ suffices to find a product. In this case, the degree of the endomorphism that we find is bounded by

$$\deg(\psi_2) \cdot \deg(\phi_1) < p^{2/3}.$$

Interestingly, this upper bound is similar to the lower bound on the degree of the smallest endomorphism of a general supersingular elliptic curve. More precisely, [24, Proposition B.5] states that every supersingular elliptic curve over $\mathbb{F}_{p^2}$ has an endomorphism of reduced norm at most $1/2p^{2/3} + 1/4$, and that the exponent $2/3$ is optimal. However, we point out that not any arbitrary endomorphism can be detected by our algorithm. In contrast, we require that an endomorphism $\alpha \in \mathrm{End}(E)$ has a decomposition $\alpha = \alpha_2 \circ \alpha_1$ with isogenies $\alpha_1 : E \to E'$ and $\alpha_2 : E' \to E$ so that $\deg(\alpha_1) + \deg(\alpha_2) = 2^n$.

The underlying assumption for the (optimistic) upper bound described above is that vertices in the neighborhood of a product behave like random vertices. This assumption is clearly too strong. In our experiments, the bound $N = \log_2(p)/2$ works well. In this case, the degree of the resulting endomorphism is upper bounded by p. For further experimental data concerning the degree of the endomorphism, we refer to Section 5.

Smoothness We also note that the degree of the isogeny (resp. endomorphism) outputted by Algorithm 7 (resp. Algorithm 6) is of the form $\deg \phi_1 \cdot (2^n - \deg \phi_1)$ and there is no reason for it to be smooth. This is in contrast to other endomorphism or isogeny finding algorithms, where the result is usually described as the composition of low-degree isogenies, and very smooth.

Below, we describe two scenarios where this is a particularly useful output:

1. Given two supersingular elliptic curves E and E', Algorithm 7 finds an isogeny (namely $\psi_2 \circ \phi_1$) of degree $\deg \phi_1 \cdot (2^n - \deg \phi_1)$ between the two curves. In particular, when $\deg \phi_1$ is a prime, the extracted isogeny might be interesting: In the new cryptosystem *PRISM* [2], the degrees of the isogenies are of the form $q \cdot (2^a - q)$ where q is a prime of a bits. To implement a forgery attack (see [28]), one needs to generate an isogeny of this particular form.
2. Given a supersingular elliptic curve E, Algorithm 6 (or Algorithm 7) outputs two isogenies with domain E (namely ϕ_1 and ϕ_2) of degree $\deg \phi_1$ and $2^n - \deg \phi_1$, respectively. When $\deg \phi_1$ or $2^n - \deg \phi_1$ are prime, our algorithm gives a method to construct an isogeny of large prime degree.

Computing the degree of the endomorphism explicitly Assume we have recovered the endomorphism $\psi_2 \circ \phi_1$ by Algorithm 6, the degree of $\psi_2 \circ \phi_1$ can be computed explicitly by computing pairings.

Consider an isogeny $\phi \colon E \to E'$ between elliptic curves, for $P_N, Q_N \in E[N]$, recall the following relation ([36, III Proposition 8.1])

$$e_N(P_N, Q_N)^{\deg \phi} = e'_N(\phi(P_N), \phi(Q_N)) \tag{3}$$

where e_N (resp. e'_N) is the N-Weil pairing on E (resp. E'). In our case, given P_N, Q_N and their images under an isogeny, we can recover $\deg \phi \mod N$ from Equation (3). We provide pseudo-code for our implementation of this method in Appendix A.3. This method requires that the endomorphism was computed using Algorithm 5 as a subroutine, i.e. the output is an endomorphism of small degree.

3.4 Finding isogenies via splittings

A small change in our algorithm FindEndomorphism allows us to find isogenies between two given elliptic curves E and E'. This is outlined in Algorithm 7. Essentially, the only difference is that we now start on the product surface $E \times E'$, and need to find a non-diagonal product isogeny $F : E \times E' \to E_1 \times E_2$ from which we can extract an isogeny $E \to E'$.

The analysis of the output of Algorithm 7 is considerably easier than that of 6 (see Subsection 3.2). There do not exist "trivial" isogenies between elliptic curves. We only need to ensure that the output is not the zero map.

Proposition 3.7. *Let $\psi_2 \circ \phi_1 : E \to E'$ be a map outputted by Algorithm 6. Then $\psi_2 \circ \phi_1$ is non-zero, in particular, it is an isogeny.*

Algorithm 7 FindIsogeny

Input: $E \not\simeq E'$ two supersingular elliptic curves over a finite field $\mathbb{F}_{p^2}$
Output: an isogeny $\phi\colon E \to E'$

1: Compute a $(2,2)$-gluing $f_1\colon E \times E' \to \mathrm{Jac}(C)$ for some genus-2 curve C ▷ Algorithm 3
2: Find a good $(2^{n-1}, 2^{n-1})$-splitting $\mathrm{Jac}(C) \to E_1 \times E_2$ for some $n > 1$. It is required that the composition is a $(2^n, 2^n)$-isogeny $F\colon E \times E' \to E_1 \times E_2$ ▷ Algorithm 4 or Algorithm 5
3: Use Kani's lemma to extract the isogeny $\psi_2 \circ \phi_1\colon E \to E'$ from the product isogeny $F = \begin{pmatrix} \phi_1 & \hat{\psi}_2 \\ -\phi_2 & \hat{\psi}_1 \end{pmatrix}$

Proof. First note that Line 3 holds because the isogeny is induced by the isogeny diamond:

$$\begin{array}{ccc} E & \xrightarrow{\phi_1} & E_1 \\ \downarrow{\scriptstyle \phi_2} & & \downarrow{\scriptstyle \psi_2} \\ E_2 & \xrightarrow{\psi_1} & E' \end{array}$$

The only subtlety is to exclude the possibility that $\psi_2 \circ \phi_1$ is trivial, i.e. the zero map. This can only happen if one of ϕ_1 or ψ_2 vanish, in other words, if the product isogeny F is a diagonal isogeny. However, this is not possible since F is a $(2^n, 2^n)$-isogeny (i.e. has kernel of rank 2), and the first step of the isogeny chain is a gluing, whereas for a diagonal $(2^n, 2^n)$-isogeny, each $(2,2)$-isogeny is diagonal as well. □

4 Reductions between different hardness assumptions

There are numerous articles studying the connection between different hardness assumptions underlying isogeny-based cryptography. For recent results on the relation between the problems, we refer to [30,15,27].

In this section, we apply the main ideas underlying Algorithms 1 and 7 to include the `Splitting` problem in the landscape of equivalent hardness assumptions. Our results are sketched in Figure 4.

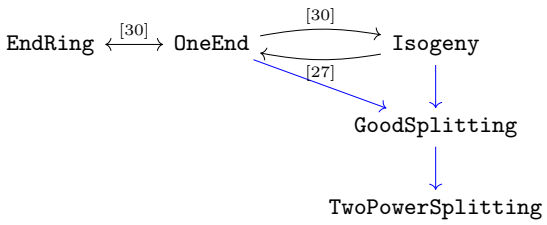

Fig. 4. Relation among different hardness assumptions in isogeny-based cryptography. The first row corresponds to a part of [27, Fig. 1]; the blue arrows are new.

4.1 Definitions of the hardness assumptions

Before we study the reductions between different problems, we provide more formal definitions.

First, we point out that whenever we ask to "find" an isogeny (including endomorphisms or splittings), we ask for an *efficient representation* in the following sense (cf. [1, Definition 3.1]):

Definition 4.1. *(Efficient representation) Let $\varphi\colon A \to B$ be an isogeny between two abelian varieties defined over a finite field $\mathbb{F}_q$. An* efficient representation *of φ with respect to a given algorithm is some data $D_\varphi \in \{0,1\}^*$ such that, on input D_φ and $P \in A(\mathbb{F}_q)$, the algorithm returns the evaluation $\varphi(P)$ in polynomial time in $\log q$ and the length of D_φ.*

In particular, we formally state the `OneEnd` and `Isogeny` problems as follows.

Problem 4.2. (`OneEnd`) Given a supersingular elliptic curve E, find an efficient representation of a non-trivial endomorphism of E.

Problem 4.3. (`Isogeny`) Given two supersingular elliptic curves E and E', find an an efficient representation of an isogeny $\phi\colon E \to E'$.

In its most general form, `Splitting` asks to find an efficient representation of an arbitrary splitting $A \to E_1 \times E_2$ on inputting a p.p. abelian surface A. We consider the following two more restrictive variants of this problem:

Problem 4.4. (`TwoPowerSplitting`) Given a principally polarized abelian surface A together with a maximal isotropic subgroup $K \subset A[2]$, find an efficient representation of a *good* $(2^{n-1}, 2^{n-1})$-splitting $F : A \to E_1 \times E_2$ for some $n > 1$. Here *good* means that $K \cap \ker(F) = \emptyset$.

We note that the meaning of *good splitting* here coincides with the terminology from Section 3. In the latter setting, the group $K \subset A[2]$ would be the kernel of the dual of the gluing isogeny.

Problem 4.5. (`GoodSplitting`) Given a principally polarized abelian surface A and a maximal m-isotropic group $K \subset A[m]$ for some integer $m \in \mathbb{N}$, find efficient representations of a good (N, N)-splitting $F : A \to E_1 \times E_2$ for some N.

Note that any (N, N)-splitting $F : A \to E_1 \times E_2$ with $\gcd(N, m) = 1$ is a solution to `GoodSplitting`. However, our definition also includes the case $\gcd(N, m) > 1$.

4.2 Reductions to `TwoPowerSplitting`

The idea for the reduction from `OneEnd` to `TwoPowerSplitting` lies at the heart of our paper. Algorithm 8 which uses only one call to the `RadicalSplitting` oracle, provides a polynomial time reduction.

Proposition 4.6. `OneEnd` *reduces to* `TwoPowerSplitting`.

Algorithm 8 Reduction from `OneEnd` to `TwoPowerSplitting`

Input: A supersingular elliptic curve E defined over $\mathbb{F}_{p^2}$
Output: A non-trivial endomorphism $\phi \in \mathrm{End}(E)$

1: Compute a $(2,2)$-gluing $f_0 : E \times E \to A$, and let $K \subset A[2]$ be the maximal isotropic subgroup defining the dual isogeny $\hat{f}_0 : A \to E \times E$.
2: Run the `TwoPowerSplitting` oracle on input (A, K) to obtain a good $(2^{n-1}, 2^{n-1})$-splitting $f_1 : A \to E_1 \times E_2$
3: Return the non-trivial endomorphism $\psi_2 \circ \phi_1$ embedded in the product isogeny $F = f_1 \circ f_0 = \begin{pmatrix} \phi_1 & \tilde{\psi}_2 \\ -\phi_2 & \tilde{\psi}_1 \end{pmatrix}$.

Proof. The reduction is sketched in Algorithm 8. This is a slightly more abstract version of Algorithm 6. The correctness of this reduction, namely that ϕ is non-scalar, was proved in Proposition 3.6. We note that the extraction of the endomorphism $\psi_2 \circ \phi_1$ from the product isogeny requires that we can evaluate F and its adjoint $\tilde{F}$ efficiently. By assumption, we know that this is true for F. Since F is a composition of 2-isogeny, one may easily compute the adjoint of each 2-isogeny, and thereby obtains an efficient representation of $\tilde{F}$ as well. Apart from this, the proof does not use any specific properties of the implementation, and immediately translates to the setting here. □

The reduction from `Isogeny` to `TwoPowerSplitting` is described in Algorithm 9. It is an abstract version of Algorithm 7, and as the previous reduction, it only requires one call to the `TwoPowerSplitting` oracle.

Algorithm 9 Reduction from `Isogeny` to `TwoPowerSplitting`

Input: Two supersingular elliptic curves E and E'
Output: An isogeny $\phi\colon E \to E'$

1: Compute a $(2,2)$-gluing $f_0 : E \times E' \to A$, and let $K \subset A[2]$ be the maximal isotropic subgroup defining the dual isogeny $\hat{f}_0 : A \to E \times E'$.
2: Run the `TwoPowerSplitting` oracle on input (A, K) to obtain a splitting $f_1 : A \to E_1 \times E_2$
3: Return the isogeny $\psi_2 \circ \phi_1$ embedded in the product isogeny
4: $F = f_1 \circ f_0 = \begin{pmatrix} \phi_1 & \tilde{\psi}_2 \\ -\phi_2 & \tilde{\psi}_1 \end{pmatrix}$.

Proposition 4.7. `Isogeny` *reduces to* `TwoPowerSplitting`.

Proof. The reduction is sketched in Algorithm 9. This is a slightly more abstract version of Algorithm 7. As in the proof of Proposition 4.6, one can show that the evaluation of the adjoint isogeny $\tilde{F}$ is efficient.

The only remaining subtlety when proving the correctness of the reduction is to make sure that $\psi_2 \circ \phi_1 \neq 0$. This follows from the fact that the product isogeny factors through a gluing, hence it cannot be diagonal. □

4.3 Generalization of the reductions

One might feel that the problem (`TwoPowerSplitting`) that we discussed is quite specific. In fact, most parts of the reductions also work in more general settings. In particular, the only restriction for extracting an elliptic curve isogeny from a product isogeny using Algorithm 14 (ExtractEndomorphism) is that the product isogeny and its adjoint can be evaluated efficiently which motivated the definition of the problem `GoodSplitting` (Problem 4.5). The more delicate part is making sure that the result is non-trivial.

Proposition 4.8. `Isogeny` *reduces to* `GoodSplitting`.

Proof. The reduction is almost the same as the reduction from `Isogeny` to `TwoPowerSplitting` sketched in Algorithm 9. The only difference occurs in Line 2, where we now call the oracle for `GoodSplitting` on input (A, K) which returns an (N, N)-splitting $f_1 : A \to E_1 \times E_2$ so that $F = f_1 \circ f_0$ is a $(2N, 2N)$-product isogeny $F : E \times E' \to E_1 \times E_2$. By assumption, we can evaluate both F and $\tilde{F}$ efficiently, hence we can extract an efficient representation of $\psi_2 \circ \phi_1$.

Moreover, as in the proof of Proposition 4.7, we find that $\psi_2 \circ \phi_1$ is non-zero since it factors through a gluing and the kernel of F has rank 2. □

We remark that the fact that the `GoodSplitting`-oracle returns a *good* splitting is essential in the proof of the reduction. This not only avoids that the returned splitting is the dual of the gluing computed in the first step, but it also ensures that other trivial cycles in the isogeny graph are avoided.

This condition becomes even more important in the next reduction, where it helps us to exclude certain special cases that would result in scalar endomorphisms. We first prove a technical lemma about diagonal isogenies appearing in the factorization of product isogenies.

Lemma 4.9. *Let $F : E \times E' \to E_1 \times E_2$ be an (N, N)-isogeny for some integer N which is induced be an isogeny diamond configuation*

$$\phi = \psi_2 \circ \phi_1 = \psi_1 \circ \phi_2,$$

with notation as in Definition 2.1.

Then $\ker(\phi_1) \cap \ker(\phi_2)$ is a cyclic group of order m for some $m \mid N$. If $m \neq 1$, then F factors through a diagonal (m, m)-isogeny $F_1 : E \times E' \to \tilde{E} \times \tilde{E}'$ for some elliptic curves $\tilde{E}, \tilde{E}'$ i.e.

$$\begin{tikzcd} E \times E' \arrow[r, "F_1"] \arrow[rr, bend right, "F"'] & \tilde{E} \times \tilde{E}' \arrow[r, "\star"] & E_1 \times E_2. \end{tikzcd}$$

If, moreover, ϕ factors through $[m]$, i.e. $\phi = [m]\phi'$ for some isogeny $\phi' : E \to E'$, then F also factors through a diagonal (m, m)-isogeny $F_2' : \tilde{E}_1 \times \tilde{E}_2 \to E_1 \times E_2$, i.e.

$$\begin{tikzcd} E \times E' \arrow[r, "\star"] \arrow[rr, bend right, "F"'] & \tilde{E}_1 \times \tilde{E}_2 \arrow[r, "F_2'"] & E_1 \times E_2. \end{tikzcd}$$

Proof. First, we note that $\ker(F)$ would be a rank-3 group if $\ker(\phi_1) \cap \ker(\phi_2)$ had rank 2. This is a contradiction to F being an (N, N)-isogeny. Hence,

$$\ker(\phi_1) \cap \ker(\phi_2) = \langle P \rangle, \quad \text{for some } P \in E[m],\ \mathrm{ord}(P) = m.$$

Necessarily $m \mid N$, and since F is an (N, N)-isogeny, $\ker(F)[m]$ is a maximal isotropic subgroup of $(E \times E')[m]$. Moreover $\langle (P, 0) \rangle \subset \ker(F)[m]$ by the definition of P. From the isotropy condition it then follows that $\ker(F)[m] = \langle (P, 0), (0, P') \rangle$ for some $P' \in E'[m]$. This is a diagonal kernel which means that F factors through a diagonal isogeny $F_1 : E \times E' \to \tilde{E} \times \tilde{E}'$.

For the second part, assume that $\phi = [m]\phi'$ for some $\phi' : E \to E'$, and consider the adjoint isogeny $\hat{F} = \begin{pmatrix} \hat{\phi}_1 & -\hat{\phi}_2 \\ \psi_2 & \psi_1 \end{pmatrix}$. Using the first part of the lemma, it suffices to show that $\ker(\hat{\phi}_1) \cap \ker(\psi_2)$ contains a point of order m. Denote $E[m] = \langle P, Q \rangle$ with P as above. Then we claim that $\phi_1(Q) \in \ker(\hat{\phi}_1) \cap \ker(\psi_2)$. First, note that $\hat{\phi}_1 \circ \phi_1(Q) = 0$, since $m \mid \deg(\phi_1)$; and $\psi_2 \circ \phi_1(Q) = 0$ since $\phi = \psi_2 \circ \phi_1$ factors through $[m]$ by our assumption. Second, note that $\mathrm{ord}(\phi_1(Q)) = m$. Otherwise, there would exist an integer $m' \mid m$ greater than 1 with $\phi_1 = [m'] \circ \phi_1'$. But then $\ker(F)[m']$ would have rank 3 which is a contradiction to our assumptions. □

Proposition 4.10. `OneEnd` *reduces to* `GoodSplitting`.

Proof. The reduction algorithm is almost the same as in the reduction from `OneEnd` to `TwoPowerSplitting` sketched in Algorithm 8. The only difference occurs in Line 2, where we now call the oracle for `GoodSplitting` on input (A, K) which returns an (N, N)-splitting $f_1 : A \to E_1 \times E_2$ so that $F = f_1 \circ f_0$ is a $(2N, 2N)$-product isogeny $F : E \times E \to E_1 \times E_2$. By assumption, we can evaluate both F and $\tilde{F}$ efficiently, hence we can extract an efficient representation of $\psi_2 \circ \phi_1$. Let us now show that the endomorphism is non-trivial. We proceed by contradiction, i.e. assume that $\psi_2 \circ \phi_1 = [M]$ for some scalar $M \in \mathbb{Z}$.

First, if $\ker(\phi_1) \cap \ker(\phi_2) = \langle P \rangle$ with $\mathrm{ord}(P) = m \neq 0$, then we apply Lemma 4.9. That is we decompose $F = F_2 \circ F_1$, where F_2 is a diagonal isogeny $F_2 : \tilde{E}_1 \times \tilde{E}_2 \to E_1 \times E_2$, and $F_1 : E \times E \to \tilde{E}_1 \times \tilde{E}_2$ is a product isogeny induced by an isogeny diamond of order N/m. It suffices to show that this isogeny diamond induces a non-scalar product isogeny. Having this in mind, we may and do assume that

$$\ker(\phi_1) \cap \ker(\phi_2) = \{0\}.$$

Now, similar to the proof of Proposition 3.6, we write

$$\phi_1 = [a] \circ \phi_1', \quad \psi_2 = [b] \circ \hat{\phi}_1'$$

and

$$\phi_2 = [b] \circ \phi_2', \quad \psi_1 = [a] \circ \hat{\phi}_2'$$

for some cyclic isogenies ϕ_1' and ϕ_2' (here we use the fact that $\ker(\phi_1) \cap \ker(\phi_2) = \{0\}$). We further remark that

$$2N = \deg(\phi_1) + \deg(\phi_2) = d \cdot (a^2 + b^2),$$

where $d = \deg(\phi_1') = \deg(\phi_2')$.

Consequently, the product isogeny is of the form

$$F = \begin{pmatrix} [a]\phi_1' & [b]\phi_1' \\ -[b]\phi_2' & [a]\phi_2' \end{pmatrix}.$$

We distinguish two cases.

If $a \equiv b \pmod 2$, then both a and b are odd (and coprime), since $\ker(\phi_1) \cap \ker(\phi_2) = \{0\}$. In this case

$$F : \begin{pmatrix} P \\ P \end{pmatrix} \mapsto \begin{pmatrix} \phi_1'([2]P) \\ \phi_2'([2]P) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{for all } P \in E[2].$$

This means $\ker(F)[2] = \{(P, P) \mid P \in E[2]\}$ which is a contradiction by Lemma 3.5.

If $a \not\equiv b \pmod 2$, assume without loss of generality that a is even and b is odd. Then

$$F : \begin{pmatrix} P \\ Q \end{pmatrix} \mapsto \begin{pmatrix} \phi_1'(Q) \\ \phi_2'(P) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \text{for } (P, Q) \in \ker(\phi_2') \times \ker(\phi_1') \subset (E \times E)[2].$$

This means that $\ker(F)[2]$ is the kernel of a diagonal $(2, 2)$-isogeny in contradiction with our construction. □

Remark 4.11. Note that in the reduction `Isogeny` to `GoodSplitting`, we did not make use of the fact that the first step is a $(2, 2)$-gluing. Indeed, one could use an arbitrary (n, n)-gluing provided that the splitting is *good* with respect to this gluing.

This is different from the reduction `OneEnd` to `GoodSplitting`. Here, the fact that the first step is a $(2, 2)$-gluing is essential in the proof. And indeed, it cannot be generalized to arbitrary gluings in an obvious way. We show this in Example 4.12.

Example 4.12. Let E be an arbitrary elliptic curve and denote $E[3] = \langle P, Q \rangle$. We consider the isogeny diamond

$$\begin{array}{ccc} E & \xrightarrow{\phi_1} & E_1 \\ \downarrow{\scriptstyle [2]\phi_2} & & \downarrow{\scriptstyle [2]\hat{\phi}_1} \\ E_2 & \xrightarrow{\hat{\phi}_2} & E \end{array}$$

where $\ker(\phi_1) = \langle P \rangle$ and $\ker(\phi_2) = \langle Q \rangle$.

This isogeny diamond induces a $(15, 15)$-product isogeny $F = f_1 \circ f_0$. Without loss of generality assume that f_0 is a $(3, 3)$-isogeny. Then f_0 is necessarily a gluing unless there exists a 2-isogeny $E \to E$.

This is an example for a nontrivial (N, N)-product isogeny with domain $E \times E$ which induces a scalar endomorphism, here $([2]\hat{\phi}_1) \circ \phi_1 = [6]$.

Remark 4.13. We have shown that computing endomorphisms in dimension 1 essentially reduces to finding splittings in dimension 2. It is natural to wonder about the relation with finding a non-trivial endomorphism in dimension 2. To make this question more precise, we recall that the endomorphism ring of a superspecial product of elliptic curves, $E_1 \times E_2$, is of the form $\begin{pmatrix} \mathrm{End}(E_1) & \mathrm{Hom}(E_2, E_1) \\ \mathrm{Hom}(E_1, E_2) & \mathrm{End}(E_2) \end{pmatrix}$.

Using a `GoodSplitting` oracle, we can find a non-trivial isogeny for each entry by the methods outlined in this article. Moreover, given an irreducible superspecial A, using the `GoodSplitting` oracle again, allows us to find $\varphi\colon A \to E \times E'$ for some E and E'. Let $\psi \in \mathrm{End}(E \times E')$ be an endomorphism found as described above, then the composition $\tilde{\varphi} \circ \psi \circ \varphi \in \mathrm{End}(A)$ is an endomorphism. This suggests a possible reduction from `OneEndDim2` to `GoodSplitting`, where `OneEndDim2` asks to find a non-trivial endomorphism of a principally polarized abelian surface. However, we note that the map $\tilde{\varphi} \circ \psi \circ \varphi$ is not necessarily an (N, N)-isogeny for some $N \in \mathbb{N}$, and it might not even be an isogeny of principally polarized abelian surfaces. More advanced techniques are necessary, to find an endomorphism of $\mathrm{End}(A)$ which is compatible with the polarization.

Another reduction related to this setting was suggested by one of the reviewers of this article: If we had an oracle to compute $\mathrm{End}(A)$ on input A, then we could use KLPT^2 [6] in order to construct a splitting $A \to E \times E'$.

Further analysis of the relation of `Splitting` with the `OneEnd` and `EndRing` problem in dimension 2 are left for future work.

5 Experimental results

We implemented our algorithms in SageMath 10.5 [38], and conducted different experiments. The main goal of the experiments is to get a better understanding on the distribution of endomorphisms embedded in 2-dimensional product isogenies.

For these experiments we use the following primes

$$p = 4099, 8219, 16411, 32771, 65539, 131111, 262147, 524347, 1048703.$$

These are the smallest primes of bit length $12, \ldots, 20$ respectively, for which the supersingular elliptic curves over $\mathbb{F}_{p^2}$ have full rational 4-torsion. This guarantees that all level-2 theta null points computed in our experiments are $\mathbb{F}_{p^2}$-rational (see [22, Example 30]). The experimental data is summarized in Figures 5 and 6.

In Figure 5 we compare our methods `FindEndomorphism` and `FindSmallEndomorphism`. Both algorithms have asymptotic running time $O(p)$, and as expected `FindEndomorphism` which computes an endomorphism of arbitrary degree is faster than `FindSmallEndomorphism` which computes an endomorphism of degree bounded by $N = \log_2(p)$. As a reference for state-of-the-art algorithms in dimension one, we use the `CycleFinding` approach from [15]. Having asymptotic complexity $O(\sqrt{p})$, the performance is clearly better than that of our new methods.

In Figure 6, we compare the degree of the endomorphism computed with different methods. We exclude the algorithm `FindEndomorphism` as it typically produces an endomorphism of degree $\approx 2^p$, and it is not clear how to determine the exact degree from the 2-dimensional representation in this case (cf. Subsection 5.2). Instead, we compare the degree of the endomorphism found with `FindSmallEndomorphism` with the smallest endomorphism embedded in a $(2^n, 2^n)$-isogeny. For comparison, we further added the degree of the endomorphism found with the 1-dimensional cycle finding method from [15]. Here, we observe that the degree of the smallest endomorphism embedded in a $(2^n, 2^n)$-product isogeny is close to the upper bound on the degree of

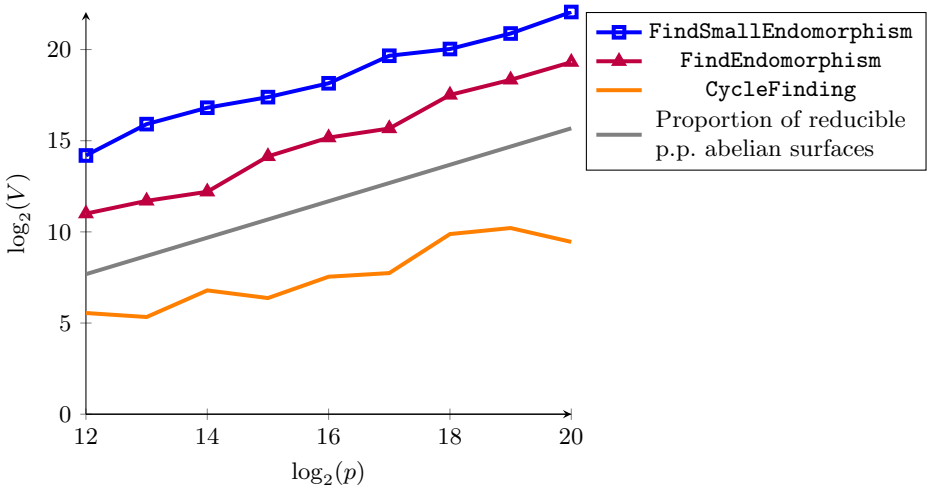

Fig. 5. Vertices visited in the isogeny graph (V) until an endomorphism is found. Comparison between the methods `FindEndomorphism`, `FindSmallEndomorphism` and `CycleFinding`.

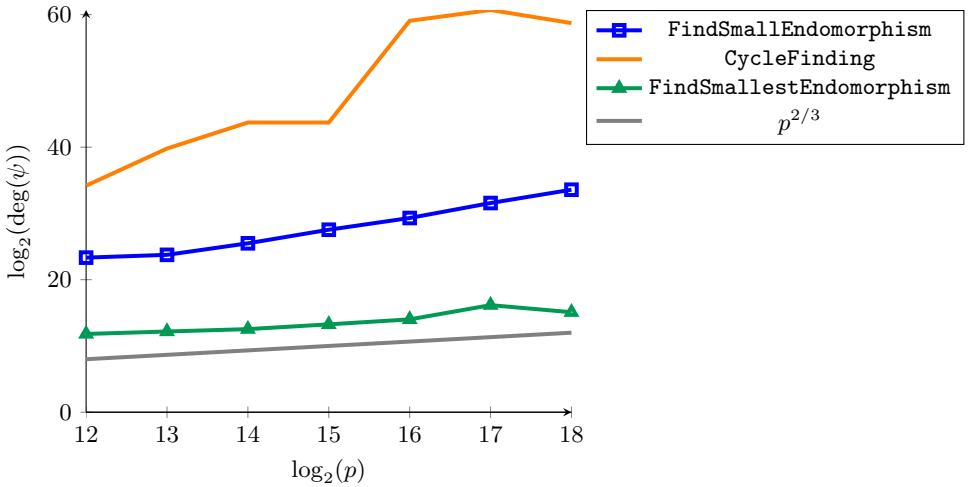

Fig. 6. Comparison of the degrees of the endomorphism, between `FindSmallEndomorphism`, `CycleFinding` and the smallest possible degree of an endomorphism embedded in a $(2^n, 2^n)$-isogeny.

the smallest endomorphism of an elliptic curve (cf. Subsection 3.3). In our experiments, the degree of the endomorphism found with our methods is typically larger, but it is still significantly smaller than the degree of the endomorphisms found with the cycle finding method.

More details including explicit running times and implementation details are provided in the remainder of this section. Experiments were run on a computer with an Intel i7-1365U processor running Ubuntu version 22.04 with SageMath version 10.5 installed. All code is made available in our GitHub repository [34]:

`https://github.com/easonmath/endomorphisms-via-splittings`

5.1 The cycle-finding algorithm

For reference, we included a comparison to one of the state-of-the-art algorithms to compute endomorphisms of elliptic curves: the cycle-finding algorithm from [15]. We briefly summarize this method here. It computes an endomorphism on input a supersingular j-invariant j_0.

1. Starting from j_0, take a random walk on the isogeny graph.
2. Repeating Step 1, stop when j_k is adjacent to j_k^p.
3. Appending $\{j_0, \cdots, j_k\}$ with $\{j_k^p, \cdots, j_0^p\}$.
4. Repeating the steps above (if necessary) to obtain another $\{j_0, \cdots, j_k', j_k'^p, \cdots, j_0^p\}$.
5. Return $\{j_0, j_1, \cdots, j_k, j_k^p, \cdots, j_0^p, j_1'^p, \cdots, j_k'^p, j_k', \cdots, j_0\}$.

If a pair of j-invariants (j, j') is a root of the modular polynomial, then so is (j^p, j'^p), which means the third step is automatic.

Remark 5.1. First, we remark that the algorithms from [15] work for ℓ-isogeny graphs where ℓ needs not be 2. In the comparison with our method, we only consider a version based on 2-isogenies. Further, we note that their finding-algorithm actually finds $j_k \in \mathbb{F}_p$, i.e., $j_k = j_k^p$. Thus, the degree of the endomorphism is always a power of 2.

In Figure 5, we compare the number of vertices visited in the isogeny graph until an endomorphism is found. This Figure is based on the values from Table 1. For each prime p, we compute the average over 10 randomly chosen elliptic curves as input.

Prime	$\log_2(p)$	FindEndomorphism	FindSmallEndomorphism	CycleFinding
4099	12	2055	18692	47
8219	13	3330	61475	40
16411	14	4712	115281	111
32771	15	18145	172265	83
65539	16	36772	291312	186
131111	17	52065	826845	214
262147	18	186309	1073583	940
524347	19	332190	1923937	1186
1048703	20	651946	4377878	697

Table 1. Numbers of visited vertices

Counting the number of inspected vertices is an implementation independent way to analyze the complexity of the algorithm. One should have in mind that in the cycle finding method each step is the computation of an isogeny in dimension 1, while in our method for each vertex, we need to compute an isogeny in dimension 2. The cost differs by a constant factor. More precisely, if one uses radical 2-isogenies than the cost of an isogeny in dimension 2 is roughly three times as high as for a one-dimensional isogeny.

Another detail to have in mind is that in the method `FindSmallEndomorphism` certain square-root computations in the evaluation of the radical isogeny formulas can be reused. This reduces the average cost of an isogeny computation at each step.

In our implementation, we observe the following running times: For a prime of bitlength $\log_2(p) \approx 12$, the algorithm `FindEndomorphism` needs about 0.27 seconds, while it takes about 88.01 seconds if $\log_2(p) = 20$. The algorithm `FindSmallEndomorphism` requires about 1.17 seconds if $\log_2(p) \approx 12$, and 628.52 seconds if $\log_2(p) \approx 20$.

5.2 Comparing the degree and smallest possible degree

In Figure 6, we provide some details on the degree of the endomorphism found with our methods. We recall that any supersingular elliptic curve over $\mathbb{F}_{p^2}$, has a non-scalar endomorphism of degree at most $p^{2/3}$. As we discussed in Subsection 3.3, our method cannot detect endomorphisms of arbitrary degree. In Figure 6, we illustrate the average degree of an endomorphism found with `FindSmallEndomorphism` for primes of different sizes. We compare this to the degree of the smallest endomorphism embedded in a $(2^n, 2^n)$-isogeny (assuming that we fixed the first gluing in the isogeny chain), and with the generic lower bound. The Figure is based on the values from Table 2.

Prime	$\log_2(p)$	d_α	$d_{\alpha,min}$
4099	12	10576473	3599
8219	13	14129172	4628
16411	14	47044451	5956
32771	15	195023255	9794
65539	16	666093523	16481
131111	17	3189167810	73185
262147	18	12866882456	34944

Table 2. Comparison: degree d_α of the endomorphism found with FindEndomorphism and the smallest possible degree $d_{\alpha,min}$ of an endomorphism embedded in a $(2^n, 2^n)$-product isogeny

In order to find the degree of an endomorphism, we use the Weil pairing (cf. Subsection 3.3). And to find the smallest good $(2^k, 2^k)$-splitting of a given p.p. abelian surface, we repeatedly use FindSmallSplitting with decreasing bound N.

References

1. Samuele Anni, Gaetan Bisson, Annamaria Iezzi, Elisa Lorenzo García, and Benjamin Wesolowski. On the computation of endomorphism rings of abelian surfaces over finite fields, 2025. arXiv: `https://arxiv.org/pdf/2503.08925v1`.
2. Andrea Basso, Giacomo Borin, Wouter Castryck, Maria Corte-Real Santos, Riccardo Invernizzi, Antonin Leroux, Luciano Maino, Frederik Vercauteren, and Benjamin Wesolowski. PRISM: Simple and Compact Identification and Signatures from Large Prime Degree Isogenies. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography – PKC 2025*, pages 300–332, Cham, 2025. Springer Nature Switzerland.
3. Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West - the fast, the small, and the safer. In *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III*, volume 15486 of *Lecture Notes in Computer Science*, pages 339–370. Springer, 2024.
4. Wouter Castryck and Thomas Decru. Multiradical isogenies. In *18th International Conference Arithmetic, Geometry, Cryptography, and Coding Theory*, volume 779, pages 57–89. American Mathematical Society, 2022.
5. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 423–447. Springer, 2023.
6. Wouter Castryck, Thomas Decru, Péter Kutas, Abel Laval, Christophe Petit, and Yan Bo Ti. KLPT^2: algebraic pathfinding in dimension two and applications. In *Annual International Cryptology Conference*, pages 167–200. Springer, 2025.
7. Wouter Castryck, Thomas Decru, and Benjamin Smith. Hash functions from superspecial genus-2 curves using Richelot isogenies. *Journal of Mathematical Cryptology*, 14(1):268–292, 2020.
8. Maria Corte-Real Santos, Craig Costello, and Sam Frengley. Efficient algorithms for the detection of (N, N)-splittings and endomorphisms. Cryptology ePrint Archive, Paper 2025/147, 2025.
9. Maria Corte-Real Santos, Craig Costello, and Jia Shi. Accelerating the Delfs–Galbraith Algorithm with Fast Subfield Root Detection. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 285–314, Cham, 2022. Springer Nature Switzerland.
10. Craig Costello and Benjamin Smith. The Supersingular Isogeny Problem in Genus 2 and Beyond. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography*, pages 151–168, Cham, 2020. Springer International Publishing.
11. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQISignHD: new dimensions in cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–32. Springer, 2024.
12. Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An algorithmic approach to (2, 2)-isogenies in the theta model and applications to isogeny-based cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 304–338. Springer, 2024.
13. Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *Des. Codes Cryptography*, 78(2):425–440, February 2016.
14. Max Duparc and Tako Boris Fouotsa. SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. In *Advances in Cryptology - ASIACRYPT 2024 -*

30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III, volume 15486 of *Lecture Notes in Computer Science*, pages 396–429. Springer, 2024.
15. Kirsten Eisenträger, Sean Hallgren, Chris Leonardi, Travis Morrison, and Jennifer Park. Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny graphs. In *Open Book Series 4*, 2020.
16. Enric Florit and Benjamin Smith. An atlas of the richelot isogeny graph. *RIMS Kôkyûroku Bessatsu*, 90:195–219, 2022.
17. Eberhard Freitag. *Siegelsche Modulfunktionen*, volume 254 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 2013.
18. Jenny Fuselier, Annamaria Iezzi, Mark Kozek, Travis Morrison, and Changningphaabi Namoijam. Computing supersingular endomorphism rings using inseparable endomorphisms. *Journal of Algebra*, 668:145–189, 2025.
19. Ernst Kani. Elliptic curves on abelian surfaces. *Manuscripta Math*, 84:199–223, 1994.
20. Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik*, 485:93–122, 1997.
21. David Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California at Berkeley, 1996.
22. Sabrina Kunzweiler, Luciano Maino, Tomoki Moriya, Christophe Petit, Giacomo Pope, Damien Robert, Miha Stopar, and Yan Bo Ti. Radical 2-isogenies and cryptographic hash functions in dimensions 1, 2 and 3. In *IACR International Conference on Public-Key Cryptography*, pages 265–299. Springer, 2025.
23. Eda Kırımlı and Chloe Martindale. The computational refined humbert invariant problem is equivalent to the computational isogeny problem. Cryptology ePrint Archive, Paper 2025/1295, 2025.
24. Jonathan Love and Dan Boneh. Supersingular curves with small non-integer endomorphisms. *ANTS*, pages 7–22, 2020.
25. Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 448–471. Springer, 2023.
26. Riccardo Salvati Manni. On the projective varieties associated with some subrings of the ring of Thetanullwerte. *Nagoya mathematical journal*, 133:71–83, 1994.
27. Arthur Herlédan Le Merdy and Benjamin Wesolowski. Unconditional foundations for supersingular isogeny-based cryptography. Cryptology ePrint Archive, Paper 2025/271, 2025.
28. Kohei Nakagawa and Hiroshi Onuki. Attacks on PRISM-id via torsion over small extension fields. Cryptology ePrint Archive, Paper 2025/1602, 2025.
29. Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren. SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. In *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part III*, volume 15486 of *Lecture Notes in Computer Science*, pages 272–303. Springer, 2024.
30. Aurel Page and Benjamin Wesolowski. The Supersingular Endomorphism Ring and One Endomorphism Problems are Equivalent. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 388–417, Cham, 2024. Springer Nature Switzerland.
31. Damien Robert. A note on optimising 2^n-isogenies in higher dimension, 2023.

32. Damien Robert. Breaking SIDH in polynomial time. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 472–503. Springer, 2023.
33. Damien Robert. Breaking SIDH in Polynomial Time. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 472–503, Cham, 2023. Springer Nature Switzerland.
34. Min-Yi Shen. Github repository: Endomorphisms via splittings, 2025.
35. Min-Yi Shen. A two-dimensinoal method for representing an endomorphism of an elliptic curve. Master's thesis, National Tsing Hua University, 2025.
36. Joseph H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate texts in mathematics. Springer New York, 2009.
37. Katsuyuki Takashima. Efficient algorithms for isogeny sequences and their cryptographic applications. In *Mathematical Modelling for Next-Generation Cryptography: CREST Crypto-Math Project*, pages 97–114. Springer, 2017.
38. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.5)*, 2025. `https://www.sagemath.org`.
39. Bert van Geemen and Gerard van der Geer. Kummer varieties and the moduli spaces of abelian varieties. *American Journal of Mathematics*, 108(3):615–641, 1986.

A Implementation details

In our implementation of the endomorphism finding algorithms, we had to adapt several algorithms from the literature. The high-level idea of our computations are described in Section 3. Here, we provide more details on the concrete modifications. We recall that our implementation is available at

`https://github.com/easonmath/endomorphisms-via-splittings`

A.1 Evaluating isogeny chains

In our implementation, isogenies are computed using radical formulas and level-2 theta coordinates. In particular, the resulting isogenies are represented by the level-2 coordinates of the domain, and codomain of the isogeny (see Figure 1). In most cases, this information is enough to evaluate the corresponding isogeny efficiently on any input (see Figure 2). However, when the domain is reducible, several subtleties appear (see Remark 2.5). In this section, we describe these subtle points in more detail, and explain the solution followed in our implementation. In particular, we describe the following algorithms

- SpecialEvaluation (Algorithm 10): This describes the evaluation of an isogeny when one of the theta null values vanishes.
- Find4Torsion (Algorithm 11): The computation of 4-torsion points lying above a specified 2-torsion point.
- GluingEvaluation (Algorithm 12): This method is based on the previous two algorithms, and describes the evaluation of the gluing isogeny at an arbitrary point.
- ChainEvaluation (Algorithm 13): This is the resulting algorithm to evaluate a $(2^n, 2^n)$-isogeny represented as a chain of theta null values.

– EvaluateEndomorphism (Algorithm 14): Given a chain of theta null values representing a product isogeny $F : E \times E \to E_1 \times E_2$, the algorithm explains the evaluation of the endomorphism represented by this chain.

Regarding the first gluing, we apply the idea in [12], which is stated in Algorithm 10: To evaluate a gluing $f\colon E \times E' \to A_1$, where $\tilde{\theta}^{A_1}(0)$ has a zero, we require a 4-torsion point T_1' lying above T_1, where $\ker(f) = \langle (a_0 : -a_1 : a_2 : -a_3), (a_0 : a_1 : -a_2 : -a_3) \rangle := \langle T_1, T_2 \rangle$; that is $2T_1' = T_1 \in \ker(f)$. The key reason is the *translation formula*, given a 2-torsion point S which corresponds to $(j, \chi) \in (\mathbb{Z}/2\mathbb{Z})^g \times (\widehat{\mathbb{Z}/2\mathbb{Z}})^g$ and any point P, we have

$$(\theta_i(P + S))_i = (\chi(i)\theta_{i+j}(P))_i. \tag{4}$$

So, writing

$$\tilde{\theta}^{A_1}(f(P)) = \left(\tilde{\theta}^{A_1}_{00}(f(P)) : \tilde{\theta}^{A_1}_{10}(f(P)) : \tilde{\theta}^{A_1}_{01}(f(P)) : \tilde{\theta}^{A_1}_{11}(f(P))\right),$$

we have

$$\begin{aligned}\mathcal{H} \circ \mathcal{S}(\theta^A(P + T_1')) &= \tilde{\theta}^{A_1}(0) \star \tilde{\theta}^{A_1}\left(f(P) + f(T_1')\right) \\ &= \tilde{\theta}^{A_1}(0) \star \left(\tilde{\theta}^{A_1}_{10}(f(P)) : \tilde{\theta}^{A_1}_{00}(f(P)) : \tilde{\theta}^{A_1}_{11}(f(P)) : \tilde{\theta}^{A_1}_{01}(f(P))\right),\end{aligned}$$

where $(x_i)_i \star (y_i)_i = (x_i y_i)_i$. The second equality comes from the translation formula because $f(T_1') \in A_1[2]$ corresponds to $((0,0), \chi_{(1,0)})$. In other words, T_1' helps us to swap the desired components and therefore one could recover the missing coordinate from the zero coordinate in $\mathcal{H} \circ \mathcal{S}(\theta^{E \times E'}(P))$. Hence, it remains to find $\theta^{E \times E'}(T_1')$ and $\theta^{E \times E'}(P + T_1')$.

Finding the four-torsion point. To find a compatible 4-torsion point T_1' on a reducible p.p. abelian surface D (not necessarily in the product structure), we first note that we know the form of $\theta^D(T_1')$ from the relation $T_1' + T_1 = -T_1'$:

$$\begin{aligned}\theta^D(-T_1') = \theta^D(T_1') &= (s_0 : s_1 : s_2 : s_3) \\ &= (s_0 : -s_1 : s_2 : -s_3)\end{aligned}$$

where the second equality comes from the translation formula (Eq. (4)). So the torsion point is of the form $(x : 0 : 1 : 0) \in \mathbb{P}^3$ for some x in the base field. It remains to solve for x from the fact $\tilde{f} \circ f(\theta^D(T_1')) = \theta^D(T_1)$. We provide the pseudo-code in Algorithm 11.

Evaluation of the gluing. Computing $\theta^{E \times E'}(P + T_1')$ from $\theta^{E \times E'}(P)$ and $\theta^{E \times E'}(T_1')$ is not possible because one needs to know $\theta^{E \times E'}(P - T_1')$ to do the differential addition. A method is to compute the addition in Weierstrass coordinates before turning into the theta model. By combining this idea with the two algorithms above, we provide the pseudo-code of the gluing-evaluation in Algorithm 12.

Notice that in Step 3, one actually recovers four valid points $(\pm T_{11}', \pm T_{12}') \in E \times E'$ from a single $\theta(T_1')$. The four ι's in the output actually correspond to the different choices in Step 3.

Algorithm 10 SpecialEvaluation as $\alpha = 0$ ([12, Algorithm 9])

Input: $\theta^{E\times E'}(P)$, $\theta^{E\times E'}(P+T_1')$ and $\tilde{\theta}^{A_1}(0) = (\alpha,\beta,\gamma,\delta)$ where $\alpha = 0$
Output: $\theta^B(f(P))$

$X_P, Y_P, 0, W_P \leftarrow \mathcal{H}\circ\mathcal{S}(\theta^{E\times E'}(P))$
$X_{PT}, Y_{PT}, 0, W_{PT} \leftarrow \mathcal{H}\circ\mathcal{S}(\theta^{E\times E'}(P+T_1'))$
$Y, Z, W \leftarrow Y_P/\beta, Z_P/\gamma, W_P/\delta$
if $Z \neq 0$ **then**
    $Z \leftarrow Z/\delta$
    $\lambda^{-1} \leftarrow Z/W_{PT}$
else
    $Z' \leftarrow Z/\gamma$
    $\lambda \leftarrow W/Z'$
end if
$X' \leftarrow Y_{PT}/\beta$
$X \leftarrow X\cdot\lambda^{-1}$
$X, Y, Z, W \leftarrow \mathcal{H}(X, Y, Z, W)$
return (X, Y, Z, W)

Algorithm 11 Find4Torsion

Input: A theta null point $(a_0 : a_1 : a_2 : a_3)$ of a reducible p.p. abelian surface (not necessarily in the product structure)
Output: The theta coordinates of a 4-torsion point lying above $(a_0 : -a_1 : a_2 : -a_3)$

Compute the automorphism μ such that $(E\times E')_\mu$ is in the product structure and $a \leftarrow \theta^{(E\times E')_\mu}$
Doubling $\mu\cdot(x,0,1,0)$ by applying the radical-2 isogeny and its adjoint. Denote the result by (X, Y, Z, W) where the coordinates is represented by the unknown x
Solve for x from $\mu^{-1}(X : Y : Z : W) = (a_0 : -a_1 : a_2 : -a_3)$ in the projective space
return $(x : 0 : 1 : 0)$

Algorithm 12 GluingEvaluation

Input: $\theta^{(E\times E')_\sigma}(0)$, $\theta^{A_1}(0)$ and $(P, Q) \in E\times E'$
Output: $\theta^{A_1}(f\circ\iota(P))$ where f is a $(2,2)$-gluing and $\iota \in \{[\pm 1]\times[\pm 1]\}$

$\theta^\sigma(T_1') \leftarrow$ Find4Torsion on $(E\times E')_\sigma$
$\theta(T_1') \leftarrow \sigma^{-1}\theta^\sigma(T_1')$
Recover (T_{11}', T_{12}') in Weierstrass coordinates from $\theta(T_1')$
Compute $(P+T_{11}', Q+T_{12}')$ and transform it to the theta model
$\theta^\sigma(P), \theta^\sigma(P+T_1') \leftarrow \sigma\theta(P), \sigma\theta(P+T_1')$
$(X, Y, Z, W) \leftarrow$ SpecialEvaluation$(\theta^\sigma(P), \theta^\sigma(P+T_1'), \tilde{\theta}^{A_1}(0))$
return (X, Y, Z, W)

Lemma A.1. *The four choices in Step 4 of Algorithm 12*

$$(P+T'_{11},Q+T'_{12}),(P-T'_{11},Q-T'_{12}),(P+T'_{11},Q-T'_{12}),(P-T'_{11},Q+T'_{12})$$

correspond to right-composition of the gluing isogeny with the four isomorphism $[\pm 1]\times[\pm 1]$.

Proof. Using the notation from Algorithm 12. Recall that we are working on a Kummer variety $E/\pm 1$ (resp. $E'/\pm 1$), so P and $-P$ (resp. Q and $-Q$) give the same theta coordinates. Therefore, for example,

$$\theta(P-T'_{11},Q+T'_{12})=\theta(-P+T_{11'},Q+T'_{12}).$$

In other words, if we say the first choice leads to $f((P,Q))$ then the last choice actually gives $f((-P,Q))$. □

Although we know the relation between the choices, there is no reason to say that the first choice yields $\theta(f((P,Q)))$, so Algorithm 12 actually compute $f\circ\iota$ for an $\iota\in\{[\pm 1]\times[\pm 1]\}$.

The overall evaluating-algorithm is stated in Algorithm 13 and Evaluation used there is simply the formula $\mathcal{H}\circ\mathcal{C}_{\tilde{b}}\circ\mathcal{H}\circ\mathcal{S}$. A subtle thing is that the effect of ι produced by GluingEvaluation can be ignored because the theta coordinates originally represent a point up to a sign.

Algorithm 13 ChainEvaluation

Input: A chain $\mathsf{NL}=(\mathsf{NL}'_0,\mathsf{NL}_0,\mathsf{NL}_1,\ldots,\mathsf{NL}_n)$ containing the theta null points of an isogeny chain as in (Eq. 1), and $\theta^{E\times E'}((P,Q))$
Output: $F(P,Q)$ in Weierstrass model where F is induced by NL
1: $\theta(F((P,Q)))\leftarrow\mathsf{GluingEvaluation}(\mathsf{NL}[1],\mathsf{NL}[2],(P,Q))$
2: **for** $i\in\{0,\cdots,\#\mathsf{NL}-3\}$ **do**
3: $\quad\theta(F((P,Q)))\leftarrow\mathsf{Evaluation}(\mathsf{NL}[i+3],\theta(F((P,Q))))$
4: **end for**
5: $\theta(F_\tau((P,Q)))\leftarrow\tau\theta(F((P,Q)))$
6: Recover the pair $F_\tau(P,Q)\in E_1\times E_2$ from its theta coordinates
7: **return** $F_\tau((P,Q))$

A.2 The adjoint isogeny

Assume that we are given an efficient representation of an isogeny as the composition of small-degree isogenies. Then we can also find an efficient representation of the adjoint isogeny. The reason is that the latter can be decomposed as a composition of small-degree isogenies as well. Furthermore the computation of the adjoints can be done step-by-step. Here we describe these computations in more detail for the isogeny chain considered in Subsection A.1.

To extract the endomorphism, we also need to evaluate the adjoint isogeny induced by the same chain. Due to the symmetry of our chain (1), the algorithms AdjointSpecialEvaluation, AdjointFind4Torsion, AdjointGluingEvaluation and AdjointChainEvaluation are quite similar to the corresponding algorithms. For simplicity, we only note three important points here.

1. To evaluate a point via the adjoint of $f\colon A \to B$, one applies the formula $\mathcal{C}_a \circ \mathcal{H} \circ \mathcal{S} \circ \mathcal{H}$ instead.
2. The auxiliary 4-torsion point S_1' in AdjointSpecialEvaluation lies above S_1 where $\ker(\tilde{f}) = \langle (a_1 : a_0 : a_3 : a_2), (a_2 : a_3 : a_0 : a_1) \rangle := \langle S_1, S_2 \rangle$. Again, one can show that $S_1' = (y : y : 1 : 1)$ for some y in the base field, and use the same strategy to find y.
3. AdjointGluingEvaluation only computes the dual-gluing up to some $\iota' \in \{[\pm 1] \times [\pm 1]\}$, and again, it does not affect the output of AdjointChainEvaluation.

Using the described subroutines, the evaluation of the endomorphism is described in Algorithm 14.

Algorithm 14 EvaluateEndomorphism

Input: A chain $\mathsf{NL} = (\mathsf{NL}_0', \mathsf{NL}_0, \mathsf{NL}_1, \ldots, \mathsf{NL}_n)$ containing the theta null points of an isogeny chain as in (Eq. 1), and $P \in E$

Output: The image of P under the endomorphism (namely $\psi_2 \circ \phi_1$) induced by the chain.

1: $(\phi_1(P), *) \leftarrow \mathsf{ChainEvaluation}(\mathsf{NL}, (P, 0_E))$
2: $(*, -\psi_2 \circ \phi_1(P)) \leftarrow \mathsf{AdjointChainEvaluation}(\mathsf{NL}, (\phi_1(P), 0_{E_2}))$
3: **return** $\psi_2 \circ \phi_1(P)$

A.3 Computing the degree of the endomorphism

Once, we found an endomorphism, we would also like to know its degree. As we mentioned in Section 3.3, we can recover the degree by evaluating the Weil pairing on pairs of points on the domain and codomain. More precisely, we use the relation given in Equation 3. We used this idea in our implementation of the experiments in Subsection 5.2. Here, we make this explicit by describing our implementation.

Algorithm 15 summarizes the idea for the ϕ_1-entry, and we note that one can easily adapt the algorithm to compute the values for the other entries.

Algorithm 15 WeilConstraint

Input: A null list inducing the $(2^s, 2^s)$-isogeny $F = \begin{pmatrix} \phi_1 & \hat{\psi}_2 \\ -\phi_2 & \hat{\psi}_1 \end{pmatrix}$ and a positive integer $N \mid p+1$

Output: r_1 where $r_1 \equiv \deg \phi_1 \mod N$

1: Generate $P_N, Q_N \in E[N]$ where $N \mid p+1$
2: Compute $F(P_N, 0)^\top = (\phi_1(P_N), \phi_2(P_N))^\top$
3: Compute $F(Q_N, 0)^\top = (\phi_1(Q_N), \phi_2(Q_N))^\top$
4: Using Equation 3 on P_N, Q_N and their images under ϕ_1, we obtain $r_1 \equiv \deg \phi_1 \mod N$
5: **return** r_1

Furthermore, since we work with theta coordinates, we can only compute the pairing up to a sign (or see Remark A.3 for another approach); more precisely,

$$e_N(\pm\phi(P_N), \pm\phi(Q_N)) = e_N(P_N, \pm Q_N)^{\deg\phi} = e_N(P_N, Q_N)^{\pm\deg\phi}$$

as we can only recover $\pm\phi(P_N)$ and $\pm\phi(Q_N)$ from their theta coordinates. This is the reason for the signs in Line 4 of Algorithm 16.

In Algorithm 16, we apply the idea discussed above in our case to find $\deg\phi_1 \deg\psi_2$ up to a multiple of $p+1$: For a supersingular elliptic curve $E/\mathbb{F}_p$, we have $E(\mathbb{F}_{p^2}) \cong \mathbb{Z}/(p+1)\mathbb{Z} \times \mathbb{Z}/(p+1)\mathbb{Z}$ and

$$(\phi + m[p+1])P = \phi(P) + m[p+1]P = \phi(P)$$

for any $m \in \mathbb{Z}$ and $P \in E(\mathbb{F}_{p^2})$.

Algorithm 16 FindDegree

Input: A null list NL inducing the $(2^s, 2^s)$-isogeny $F = \begin{pmatrix} \phi_1 & \hat{\psi}_2 \\ -\phi_2 & \hat{\psi}_1 \end{pmatrix} : E \times E \to E_1 \times E_2$

Output: $\deg\psi_2 \deg\phi_1$

1: Generate $P \in E[p+1]$ and fix $N \mid p+1$
2: Compute $[\deg\phi_1](P)$ from Eq. (2), and compute $[\deg\phi_2](P)$ similarly
3: $r_1 \leftarrow$ `WeilConstraint`(NL)
4: Solve $xP = [\deg\phi_1](P)$ for one x where $x \equiv \pm r_1 \mod N$. If $[2^s - x](P) = [\deg\phi_2](P)$ ruturn $x(2^s - x)$; otherwise, find another x

Remark A.2. In general, given an endomorphism η, one way to compute $\deg\eta$ is by considering field extensions. Over a sufficiently large extension, one gains access to N-torsion points for larger N. With this additional information, one can find the explicit $\deg\eta$ from Equation (3).

Remark A.3. To address the sign issue, one may additionally compute $\theta(\phi(P_N + Q_N))$ and recover $\pm\phi(P_N + Q_N)$ from it. By determining the signs of $\phi(P_N), \phi(Q_N)$ and $\phi(P_N + Q_N)$ such that $\phi(P_N) + \phi(Q_N) - \phi(P_N + Q_N) = 0$, one can identify a correct combination.