Isogeny interpolation and the computation of isogenies from higher dimensional representations

David Jao and Jeanne Laflamme

Department of Combinatorics & Optimization, University of Waterloo
200 University Ave. W, Waterloo ON N2L 3G1, Canada
{djao,jmlaflam}@uwaterloo.ca

Abstract. The Supersingular Isogeny Diffie-Hellman (SIDH) scheme is a public key cryptosystem that was submitted to the National Institute of Standards and Technology's competition for the standardization of post-quantum cryptography protocols. The private key in SIDH consists of an isogeny whose degree is a prime power. In July 2022, Castryck and Decru discovered an attack that completely breaks the scheme by recovering Bob's secret key, using isogenies between higher dimensional abelian varieties to interpolate and reconstruct the isogenies comprising the SIDH private key. The original attack applies in theory to any prime power degree, but the implementation accompanying the original attack required one of the SIDH keys involved in a key exchange to have degree equal to a power of 2. An implementation of the power of 3 case was published subsequently by Decru and Kunzweiler. However, despite the passage of several years, nobody has published any implementations for prime powers other than 2 or 3, and for good reason — the necessary higher dimensional isogeny computations rapidly become more complicated as the base prime increases. In this paper, we provide for the first time a fully general isogeny interpolation implementation that works for any choice of base prime, and provide timing benchmarks for various combinations of SIDH base prime pairs. We remark that the technique of isogeny interpolation now has constructive applications as well as destructive applications, and that our methods may open the door to increased flexibility in constructing isogeny-based digital signatures and cryptosystems.

1 Introduction

Supersingular Isogeny Diffie-Hellman (SIDH) is a key-exchange protocol based on walks on the isogeny graph of supersingular elliptic curves. It was first proposed in 2011 by Jao and De Feo [11], and was submitted to the United States National Institute of Standards and Technology's (NIST) competition for standardization of post-quantum cryptography protocols. The protocol advanced to

the fourth round of the competition as an alternate candidate. However, in July 2022, Castryck and Decru [5] found a devastating attack against the protocol that allowed to recover Bob's secret key in a few hours on a laptop. Later improvements by others, notably Oudompheng [15], made it possible for the attack to run in a few seconds.

In the SIDH protocol, Alice's and Bob's secret isogenies are usually taken to have degree 2^a and 3^b respectively. In order to recover Bob's secret isogeny, the Castryck-Decru attack requires computing a degree 2^a isogeny in a 2-dimensional abelian variety. Formulas by Richelot [19] provide an extremely efficient method to compute such isogenies. If instead, we wanted to recover Alice's secret isogeny, we would need to compute a degree 3^b isogeny in dimension 2. The necessary formulas were published by Bruin et al. [3] and implemented by Decru and Kunzweiler [9,14].

One can naturally ask about how to implement the Castryck-Decru attack when Alice and Bob's secret isogenies have degrees equal to powers of primes other than 2 or 3. Implementing the attack in this case requires computing a degree ℓ^e isogeny in dimension 2 for $\ell > 3$. More specifically, one needs to compute an isogeny from a product of elliptic curves to a Jacobian of a hyperelliptic curve (the so-called "glue" isogeny), followed by a chain of isogenies between Jacobians of hyperelliptic curves, and finally an isogeny from a Jacobian back to a product of elliptic curves (the "split" isogeny). Formulas for 2-dimensional degree ℓ^e isogenies between Jacobians of hyperelliptic curves have been developed by Cosset and Robert [7], and these formulas are *mostly* implemented in a Magma package called `Avisogenies` [16]. However, explicit formulas for the necessary glue and split isogenies are not available in any prior works.

Contribution. We present the first full working implementation of isogeny interpolation involving the mapping of a point through a glue-and-split chain of 2-dimensional isogenies of degree $\ell > 3$. To do this, we add conversion formulas for converting from Mumford to theta coordinates in the glue and split steps to the `Avisogenies` package. We also fix some bugs in the `ImagePoint` function, which was flagged as untested in the source code and, indeed, contains some bugs. Finally, we extend the `ImagePoint` function so that it takes theta coordinates as input, in order to be able to use it in the glue and split cases. We then demonstrate that the computation of this isogeny chain can be used to recover Alice's secret key in the SIDH protocol when Bob's base prime is $\ell_B > 3$.

Related work. Paradoxically, the technique of representing an isogeny using an SIDH public key turns out to be fairly efficient compared to other possible approaches, and forms a central building block in the next generation of isogeny-based cryptosystems designed to succeed SIDH. Robert [18] has coined the term *higher dimensional representation* or *HD representation* to denote such isogeny representations, and the process of computing an isogeny given its HD representation is known as *isogeny interpolation*, with the idea being that the pairs

of input-output values that comprise an HD representation can be interpolated to produce an isogeny function that matches these known values. Our results can be viewed as an implementation of isogeny interpolation in the case where the known isogeny values lie in the ℓ^e torsion subgroup. Notably, isogeny interpolation is used in isogeny-based signature schemes such as SQIsignHD [8] and SQIsign2D-West [1], as well as the FESTA isogeny-based cryptosystem [2]. Although it is unlikely that the $\ell > 3$ case will ever outperform the extremely efficient Richelot isogeny formulas for $\ell = 2$, it is possible that future protocols may be able to provide more advanced functionality or benefit in other yet to be determined ways using the added flexibility provided by more base primes.

Outline. In Section 2, we present some mathematical preliminaries along with a description of the SIDH protocol and the Castryck-Decru attack. In Section 3, we introduce theta coordinates and the `Avisogenies` package. We then chronicle all the missing steps for implementing the attack with more general primes and explain how each one was done. In Section 4, we present the magma code implementing the attack. Finally, Section 5 gives timing benchmarks using different parameters and proposes future work.

2 Preliminaries

2.1 Abelian Varieties of Genus 2

Elliptic curves are abelian varieties of genus one and dimension one. The following definition generalizes this concept to higher genera.

Definition 1 (Hyperelliptic Curve). *A **hyperelliptic curve** C of genus g defined over a field K is an equation of the form*

$$C : y^2 + h(x)y = f(x),$$

where $h, f \in K[x]$ with $\deg(h) \leq g$ and $2g + 1 \leq \deg(f) \leq 2g + 2$.

If the characteristic of K is different from 2, we can always complete the square on the left to eliminate h. Since we will be working with fields $\mathbb{F}_{p^2}$ for large primes p, the characteristic condition always holds, and so we only consider hyperelliptic curves of the form $y^2 = f(x)$. We will also mostly be restricting ourselves to the case $g = 2$ from now on.

While the set of points on hyperelliptic curves of genus 2 doesn't form a group directly as it did with elliptic curves, we can construct divisors and the Jacobian of any hyperelliptic curve C in exactly the same way as one does for elliptic curves, and work with the group $\mathrm{Jac}(C)$ instead. It turns out that this Jacobian has the structure of a 2-dimensional abelian variety. In the case of hyperelliptic

curves of genus 2, every element in the Jacobian has a unique representative of the form

$$(P) + (Q) - 2(\infty),$$

where P and Q are points on the curve. Divisors in this form are referred to as ***reduced divisors***.

Mumford coordinates give a practical way to represent divisors on the Jacobian of hyperelliptic curves. Write $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. Then the Mumford representation of D is

$$D = [u(x), v(x)],$$

where $u(x) = (x - x_1)(x - x_2)$ and $v(x)$ is the line such that $v(x_1) = y_1$ and $v(x_2) = y_2$. Cantor's algorithm [4] gives a way to efficiently add divisors when they are written in this form.

We can also construct abelian varieties of dimension 2 in a different way. Namely, we can take the product group of two elliptic curves $E_1 \times E_2$. The set of points (P_1, P_2) on $E_1 \times E_2$ forms a group. Every abelian variety of dimension 2 is either the Jacobian of a hyperelliptic curve or the product of two elliptic curves.

2.2 Isogenies in Dimension 2

We are interested in isogenies between abelian varieties in dimension 2. Specifically, we are interested in isogenies of the form

$$\Phi \colon E_1 \times E_2 \to E_3 \times E_4$$

that go from a product of two elliptic curves to the product of two elliptic curves. Such isogenies were characterized by Kani's theorem [12]. They can be written in matrix form

$$\Phi = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

where $\alpha \colon E_1 \to E_3$, $\beta \colon E_2 \to E_3$, $\gamma \colon E_1 \to E_4$ and $\delta \colon E_2 \to E_4$ are 1-dimensional isogenies of elliptic curves. All our isogenies are separable, and so the number of points in $\ker \Phi$ determines its degree. The degree of dimension 2 isogenies is often written as (d, d) to emphasize the fact that we are working in higher dimension. We would like to have isogeny formulas analogous to Vélu's formulas [20] that allow us to compute the codomain of dimension 2 isogenies from their kernel.

By factoring d, we can break up Φ into a composition of smaller isogenies of prime degree. When $d = \ell^e$ for some prime ℓ, we call this composition an (ℓ, ℓ)-isogeny chain. When working over $\mathbb{F}_{p^2}$ for large p, it is overwhelmingly likely that all of the middle steps of the chain will be between Jacobians of hyperelliptic

curves. Therefore, to map a point through the chain, we first need to map it through a "glue" isogeny

$$\phi_{gl} \colon E_1 \times E_2 \to \mathrm{Jac}(H_{gl})$$

and then through a chain of $e-2$ isogenies between Jacobians

$$\phi \colon \mathrm{Jac}(C) \to \mathrm{Jac}(C')$$

and finally, through a "split" isogeny:

$$\phi_{sp} \colon \mathrm{Jac}(H_{sp}) \to E_3 \times E_4.$$

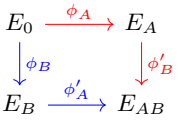
Formulas due to Richelot [19] address the three cases when $\ell = 2$. When $\ell = 3$, explicit formulas have also been found [3,9]. For general ℓ, Cosset and Robert [7] have developed algorithms that make use of theta coordinates, although complete formulas are only given for the isogenies in the middle of the chain, that go between Jacobians. These are the formulas we will be working with.

2.3 Supersingular Isogeny Diffie-Hellman

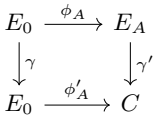
In SIDH, Alice and Bob choose two integers a and b and two small primes ℓ_A and ℓ_B such that $p = \ell_A^a \ell_B^b - 1$ is a large prime. The small primes are usually taken to be $\ell_A = 2$ and $\ell_B = 3$ to maximize the speed of the computations. However, we are interested in the case where larger values of ℓ_A and ℓ_B are chosen. From here, Alice and Bob publicly agree on a starting supersingular elliptic curve E_0 which is defined over $\mathbb{F}_{p^2}$. A common choice is $E_0 : y^2 = x^3 + 6x^2 + x$. Alice and Bob then must publicly agree on a basis $\langle P_A, Q_A \rangle$ and $\langle P_B, Q_B \rangle$ for the ℓ_A^a and ℓ_B^b torsion on E_0 respectively. This completes the initial setup.

To exchange a key, Alice and Bob start by choosing a secret integer sk_A and sk_B. Alice then computes the codomain E_A of the ℓ_A^a-isogeny ϕ_A whose kernel is generated by $P_A + sk_A Q_A$. She then transmits E_A to Bob along with the images $\phi_A(P_B)$ and $\phi_A(Q_B)$ of the generators of the ℓ_B^b-torsion under the isogeny. Bob performs an analogous procedure: he chooses a secret integer sk_B and constructs an isogeny $\phi_B \colon E_0 \to E_B$ whose kernel is $P_B + sk_B Q_B$. He then transmits $E_B, \phi_B(P_A)$ and $\phi_B(Q_A)$ to Alice. With this information, Alice computes the codomain E_{AB} of the isogeny $\phi'_A \colon E_B \to E_{BA}$ whose kernel is generated by $\phi_B(P_A) + sk_A \phi_B(Q_A)$. Bob similarly calculates the codomain E_{AB} of the isogeny $\phi'_B \colon E_A \to E_{AB}$ whose kernel is generated by $\phi_A(P_B) + sk_B \phi_A(Q_B)$. Alice and Bob thereby obtain isomorphic curves $E_{AB} \simeq E_{BA}$, so they can use the j-invariant of these curves as their shared secret key. The key exchange protocol is summarized in Figure 1a.

For SIDH to be secure, it is necessary, but not sufficient, to assume that the generic isogeny problem is hard. This hardness assumption states that, given two isogenous elliptic curves E_1 and E_2, it is computationally infeasible to recover



(a) SIDH key exchange diagram



(b) Isogeny Square from Kani's theorem

Fig. 1: Isogeny commutative diagrams

the kernel of an isogeny between them. However, in the SIDH protocol, more information than just the codomain of the isogenies is transmitted, namely, the image of the isogeny ϕ_A (resp. ϕ_B) on the ℓ_A^a (resp. ℓ_B^b) torsion. It was conjectured for a long time that the transmission of this extra information does not impact the security of the scheme. However, Castryck and Decru [5] showed that this information can be exploited to break the scheme completely.

2.4 Castryck-Decru Attack

In what follows, we describe how to recover Alice's isogeny ϕ_A. Since we are working with arbitrary ℓ_A and ℓ_B, this choice does not affect the generality of the attack. The breakthrough idea of Castryck and Decru is to look at higher dimension isogeny chains that start and end on a product of elliptic curves, which were characterized by Kani [12]. The goal is to construct an isogeny chain of abelian varieties for which the kernel can be calculated from the torsion point information and where one of the components gives us the image of the dual isogeny $\hat{\phi}_A$ of ϕ_A on any point of E_A, when mapping a suitable point through the chain. Once this work is done, we can compute $\hat{\phi}_A$ on generators P'_A and Q'_A of the ℓ_A^a-torsion on E_A. The kernel of ϕ_A is then straightforward to find, as it is generated by either of the points $\hat{\phi}_A(P'_A)$ or $\hat{\phi}_A(Q'_A)$. Castryck and Decru originally devised this attack using 2-dimensional isogeny chains, but Robert [17] later generalized their idea to dimensions 4 and 8 which allow for more flexibility in the starting curve. We will be working only with the dimension 2 case.

To construct the desired isogeny chain, we need an endomorphism γ on E_0 which has degree $c = \ell_B^b - \ell_A^a$. The purpose of choosing γ in this way is to ensure that the isogeny chain meets the criteria for Kani's theorem so that the chain indeed splits at the end. Given γ, we build the isogeny square in Figure 1b.

The dimension 2 isogeny chain forms an isogeny

$$\Phi \colon E_0 \times E_A \to E_0 \times C$$

where

$$\Phi = \begin{pmatrix} \hat{\gamma} & \hat{\phi}_A \\ -\phi'_A & \gamma' \end{pmatrix}$$

The kernel of Φ is given by

$$\langle (\ell_A^a P_B, -\phi_A(\hat{\gamma}(P_B))), (\ell_A^a Q_B, -\phi_A(\hat{\gamma}(Q_B))) \rangle,$$

where P_B and Q_B are generators for the ℓ_B^b-torsion on E_0 as in Section 2.3. If γ can be computed efficiently, this kernel can be obtained directly from the public information passed on in the SIDH protocol. Furthermore, we have that

$$\Phi(\infty, P) = (\hat{\phi}_A(P), \gamma'(P))$$

for any $P \in E_A(\mathbb{F}_{p^2})$, and therefore computing this chain on (∞, P'_A) and (∞, Q'_A) will give us $\hat{\phi}_A(P'_A)$ and $\hat{\phi}_A(Q'_A)$ as desired.

The first hurdle when implementing this attack is to find a suitable endomorphism γ which can easily be computed on any point of E_0. When E_0 is $y^2 = x^3 + 6x^2 + x$, we can take $\gamma = [u] + 2i[v]$ where $2i = \hat{\rho} \circ \iota \circ \rho$ with ρ a 2-isogeny connecting E_0 to $E'_0 : y^2 = x^3 + x$ and ι is the well-known endomorphism

$$\begin{aligned} \iota &\colon E'_0 \to E'_0 \\ (x, y) &\mapsto (-x, iy). \end{aligned}$$

With this choice, we have that $\deg(\gamma) = u^2 + 4v^2$ and so we must choose u and v so that $\ell_B^b - \ell_A^a = u^2 + 4v^2$.

The resulting isogeny chain has degree (ℓ_B^b, ℓ_B^b) and can be broken down into one (ℓ_B, ℓ_B) gluing step, $b - 2$ (ℓ_B, ℓ_B)-isogenies between Jacobians of hyperelliptic curves and one (ℓ_B, ℓ_B) splitting step. In the case where $\ell_B = 2$, the chain can be computed using formulas by Richelot which is what was done in the original implementation by Castryck and Decru. The case where $\ell_B = 3$ has also been implemented in [9].

3 Computing (ℓ, ℓ)-isogeny glue-and-split chains

In order to be able to generalize the Castryck-Decru attack to primes ℓ_A and ℓ_B that are different from 2 and 3, we need to be able to compute a chain of 2-dimensional (ℓ, ℓ)-isogenies that starts with a gluing step and ends with a splitting step for an arbitrary prime ℓ. Cosset and Robert [7] have devised algorithms to compute (ℓ, ℓ)-isogenies between abelian varieties of dimension g that make use of theta coordinates. These algorithms are implemented in a Magma package called `Avisogenies`.

3.1 Theta Coordinates and Mumford Coordinates

Theta coordinates are derived from the Riemann theta function. This function is defined on $\mathbb{C}^g \times \mathcal{H}_g$ where $\mathcal{H}_g$ is the *Siegel upper-half plane*. A matrix Ω belongs to $\mathcal{H}_g$ if and only if it is symmetric and its imaginary part is positive definite.

Definition 2 (Riemann theta function). *Let $\Omega \in \mathcal{H}_g$ and $z \in \mathbb{C}^g$. The* ***Riemann theta function*** *is defined as*

$$\theta(z, \Omega) = \sum_{n \in \mathbb{Z}_g} \exp(i\pi n^T \Omega n + 2i\pi n^T z).$$

Theta coordinates give a way to embed abelian varieties of dimension g in the projective space $\mathbb{P}^{n^g-1}$. This technique works regardless of which type of abelian variety we have. Therefore, in dimension 2, it gives us a common system of coordinates to work with both Jacobians of hyperelliptic curves and products of elliptic curves. The integer n in $\mathbb{P}^{n^g-1}$ is the *level* of the theta functions from which the coordinates originate. The Riemann theta function can be generalized by adding characteristics $a, b \in \mathbb{Q}^g$ to it. Depending on its characteristic, a theta function will satisfy a recurrence relation involving an integer n. This integer is referred to as the level of the function. For a fixed matrix Ω, the theta functions of characteristic $b \in \mathbb{Q}^g$ are defined as

$$\begin{aligned} \theta_b(\cdot, \Omega) \colon \mathbb{C}^g &\to \mathbb{C} \\ z &\mapsto \sum_{m \in \mathbb{Z}^g} \exp\left(i\pi \left(m^T \Omega m + 2m^T (z + b/2)\right)\right). \end{aligned}$$

More generally, the theta functions of characteristic $a, b \in \mathbb{Q}^g$ are defined as

$$\theta_{a,b}(z, \Omega) = \sum_{m \in \mathbb{Z}^g} \exp\left[i\pi \left(\left(m + \frac{a}{2}\right)^T \Omega \left(m + \frac{a}{2}\right) + 2\left(m + \frac{a}{2}\right)^T \left(z + \frac{b}{2}\right)\right)\right].$$

Theta functions of level n form a vector space of dimension n^g. To embed an abelian variety in projective space, we need a basis for that vector space. Different choices of bases give different embeddings. Therefore, for each abelian variety, there exist several different coordinates that can be used, depending on the level of theta functions and the basis chosen. We mainly work with theta functions of level 2 and dimension 2 and therefore we will be using $\mathbb{P}^3$.

The original Riemann theta functions were defined over the complex numbers. Abelian varieties of dimension g over $\mathbb{C}$ can be represented as points on $\mathbb{C}^g$ modulo a lattice Λ_Ω. This lattice can be written as $\Omega\mathbb{Z}^g \times \mathbb{Z}^g$ where Ω is a $g \times g$ matrix in the Siegel upper half-plane $\mathcal{H}_g$. However, in our case we are working with abelian varieties defined over a finite field $\mathbb{F}_{p^2}$ and so we will work with the analogous theta functions defined over $\mathbb{F}_{p^2}$.

A basis for the vector space of theta functions of level 2 when $g = 2$ is given by the functions whose characteristics b are $(0,0), (0,1), (1,0)$ and $(1,1)$:

$$\mathcal{F}_2(2) = \{\theta_{00}(\cdot, \Omega), \theta_{01}(\cdot, \Omega), \theta_{10}(\cdot, \Omega), \theta_{11}(\cdot, \Omega)\}.$$

This is the basis we will be mainly working with. We also need to work with genus 1 as an intermediate step in the conversion of elements on the product of

elliptic curves into theta coordinates. When $g = 1$, a basis is given by the theta functions of characteristic 0 and 1:

$$\mathcal{F}_2(1) = \{\theta_0(\cdot, \Omega), \theta_1(\cdot, \Omega)\}.$$

Starting with a divisor D given in Mumford coordinates on the Jacobian of a hyperelliptic curve C of genus 2 defined over $\mathbb{F}_{p^2}$, we would like to use the basis $\mathcal{F}_2(2)$ to map D to a point on $\mathbb{P}^3$. There exists a one-to-one correspondence between points on $\mathrm{Jac}(C)$ and elements on $(\mathbb{F}_{p^2})^2$ modulo a lattice $\Lambda_\Omega = \Omega\mathbb{Z}^2 + \mathbb{Z}^2$. We first map D to a point $z \in (\mathbb{F}_{p^2})^2$ such that $z \bmod \Lambda_\Omega$ is the point on $(\mathbb{F}_{p^2})^2/\Lambda_\Omega$ corresponding to D on $\mathrm{Jac}(C)$. We can then map z to

$$\theta^D = (\theta_{00}(z, \Omega) : \theta_{01}(z, \Omega) : \theta_{10}(z, \Omega) : \theta_{11}(z, \Omega)) \in \mathbb{P}^3(\mathbb{F}_{p^2}).$$

We refer to the point obtained on the projective space as a theta point. This mapping gives a correspondence between Divisors on $\mathrm{Jac}(C)$ and points on $\mathbb{P}^3(\mathbb{F}_{p^2})$. A more detailed introduction to theta functions and a proof of the validity of this correspondence is given in [6, Chapter 3]. We can proceed analogously to obtain the theta coordinates of a point P_1 on an elliptic curve E_1. These are given by

$$\theta^{P_1} = (\theta_0(z_1, \Omega_1) : \theta_1(z_1, \Omega_1)) \in \mathbb{P}^1(\mathbb{F}_{p^2})$$

where $\Lambda_{\Omega_1} = \Omega_1\mathbb{Z} + \mathbb{Z}$ is a lattice such that $\mathbb{F}_{p^2}/\Lambda_{\Omega_1}$ is in one-to-one correspondence with E_1 and z is a point on $\mathbb{F}_{p^2}$ which corresponds to P when modded out by Λ_{Ω_1}. If we also have the theta coordinates

$$\theta^{P_2} = (\theta_0(z_2, \Omega_2) : \theta_1(z_2, \Omega_2)) \in \mathbb{P}^1(\mathbb{F}_{p^2})$$

of a point P_2 on another elliptic curve E_2, then the theta coordinates of the point (P_1, P_2) on the variety $E_1 \times E_2$ are given by

$$\begin{aligned}\theta^{(P_1,P_2)} = \big(&\theta_0(z_1, \Omega_1)\theta_0(z_2, \Omega_2) : \theta_0(z_1, \Omega_1)\theta_1(z_2, \Omega_2) : \\ &\theta_1(z_1, \Omega_1)\theta_0(z_2, \Omega_2) : \theta_1(z_1, \Omega_1)\theta_1(z_2, \Omega_2)\big) \in \mathbb{P}(\mathbb{F}_{p^2})^3.\end{aligned}$$

The precise formulas to convert between Mumford coordinates and theta coordinates are given in the appendix of [7] and implemented in `Avisogenies`. We abbreviate the coordinate $\theta_{ij}(z, \Omega)$ as θ_{ij}^D and similarly, $\theta_i(z_j, \Omega_j)$ will be abbreviated as $\theta_i^{P_j}$. We can also represent $A = \mathrm{Jac}(C)$ itself using theta coordinates by computing

$$\theta^A = (\theta_{00}(0, \Omega) : \theta_{01}(0, \Omega) : \theta_{10}(0, \Omega) : \theta_{11}(0, \Omega)) \in \mathbb{P}^3.$$

This point is referred to as the theta null point of A and its coordinates correspond to the theta coordinates of the identity element on A. We abbreviate the coordinate $\theta_{ij}(0, \Omega)$ as θ_{ij}^A.

In Section 3.4, we also need to make use briefly of theta functions of level 4 using the basis $\mathcal{F}_{(2,2)}$. This basis includes theta functions with a non-zero characteristic $a \in \mathbb{Q}^2$. The basis is given by:

$$\begin{aligned}\mathcal{F}_{(2,2)}(2) = \big\{&\theta_{00,00}(\cdot,\Omega), \theta_{00,01}(\cdot,\Omega), \theta_{00,10}(\cdot,\Omega), \theta_{00,11}(\cdot,\Omega)\\ &\theta_{01,00}(\cdot,\Omega), \theta_{01,01}(\cdot,\Omega), \theta_{01,10}(\cdot,\Omega), \theta_{01,11}(\cdot,\Omega)\\ &\theta_{10,00}(\cdot,\Omega), \theta_{10,01}(\cdot,\Omega), \theta_{10,10}(\cdot,\Omega), \theta_{10,11}(\cdot,\Omega)\\ &\theta_{11,00}(\cdot,\Omega), \theta_{11,01}(\cdot,\Omega), \theta_{11,10}(\cdot,\Omega), \theta_{11,11}(\cdot,\Omega)\big\}.\end{aligned}$$

These functions are sometimes referred to as level $(2,2)$ because of the choice of basis. These functions are typically numbered using indices 1 to 16. Different authors have used different numberings in the past. To avoid confusion, we will stick to labelling the coordinates using the four binary digits. Some of the numberings used by other authors are listed in [6, §3.1.2].

3.2 The `Avisogenies` Package

`Avisogenies` is a Magma package developed by Damien Robert and Romain Cosset. It contains implementations of algorithms to compute 2-dimensional (ℓ,ℓ)-isogenies in theta coordinates for an arbitrary prime ℓ. Currently, it can be used to compute the theta null point of the codomain of such an isogeny by using the function `IsogenieG2Theta` in the file `isogenie.m`. It also contains implementations of the conversion between Mumford coordinates and theta coordinates in the functions `MumfordToLevel2ThetaPoint` and `Level2ThetaPointToMumford`. These functions can be used for any dimension and so they can be used to convert an elliptic curve point or a divisor on a hyperelliptic curve into theta coordinates. They can be found in the file `morphisms.m`

The package also contains conversions between a theta null point and its corresponding hyperelliptic curve of genus two. These conversions are implemented in the functions `theta_point_from_ros` and `theta_null_from_rosenhain` in the file `rosenhain.m`. The functions make use of the Rosenhain form of a curve. A hyperelliptic curve of genus 2 is in Rosenhain form if it has the form

$$y^2 = x(x-1)(x-\lambda)(x-\mu)(x-\nu).$$

The function `RosenhainForm` allows one to convert a curve to its Rosenhain form, which must be done before computing its theta null point.

Theta coordinates can also be used to represent points on a variety $A = E_1 \times E_2$ that is the product of two elliptic curves. The (ℓ,ℓ)-isogeny formulas that are implemented in `Avisogenies` work regardless of whether we are working with the theta coordinates of a product of elliptic curves or of the Jacobian of a hyperelliptic curve. However, the `Avisogenies` package has no implementation of the conversion functions between points on the product of elliptic curves and

theta coordinates, which is necessary for the gluing and splitting steps of the isogeny chain.

In order to map points through the chain, we need a function that computes the image of a point given the kernel of an isogeny. The package `Avisogenies` contains an untested function `ImagePoint`, which takes as input a divisor in Mumford coordinates on the Jacobian of a genus 2 curve, as well as the kernel of an isogeny in Mumford coordinates and returns the the image of the divisor under that isogeny as well as the codomain curve. When testing this function, we obtained an error and so some bugs had to be fixed before it could be used. We also needed to reorganize it so that it could be used for the glue and split cases as well.

In summary, in order to be able to use `Avisogenies` to compute an (ℓ, ℓ)-isogeny glue and split chain, we made the following changes:

- We implemented the conversion from an elliptic curve to theta null points.
- We mplemented the conversion from theta null points of two elliptic curves E_1 and E_2 to theta null points of the product $E_1 \times E_2$.
- We implemented the conversion from a point on a product of elliptic curves to theta coordinates.
- We fixed the `ImagePoint` function and reorganized it to use theta coordinates for input and output, so that it can be used for the glue and split cases.
- We implemented the conversion from theta null points of the product of two elliptic curves to theta null points of each elliptic curve.
- We implemented the conversion from theta null points to an elliptic curve.
- We implemented the conversion from the theta coordinates of a point (P, Q) on the product of two elliptic curves to the points P and Q.

3.3 Computing the Image of a Point

Algorithm 4.5 in [7] describes how to compute the image of a point using theta coordinates. It is implemented in the file `Image.m` of the `Avisogenies package`. However, the implementation was marked in the code as untested and contained some bugs.

The first step of the algorithm is to find an integer matrix F such that $F^T F = \ell I$. To do this, the authors first write ℓ as a sum of two squares $\ell = a^2 + b^2$, or four squares $\ell = a^2 + b^2 + c^2 + d^2$ if two is not possible. Then they take F to be either

$$F = \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \qquad \text{or} \qquad F = \begin{pmatrix} a & b & c & d \\ -b & a & -d & c \\ -c & d & a & -b \\ -d & -c & b & a \end{pmatrix}$$

Here we describe the 4×4 case, but similar remarks apply to the 2×2 case. In step 4 of Algorithm 4.5 of [7], we need a vector (m_1, m_2, m_3, m_4) such that

$$\begin{pmatrix} \ell & 0 & 0 & 0 \end{pmatrix} F^{-1} = \begin{pmatrix} m_1 & m_2 & m_3 & m_4 \end{pmatrix}.$$

The authors take this vector to be (a, b, c, d). However it doesn't satisfy the equation since

$$\begin{pmatrix} \ell & 0 & 0 & 0 \end{pmatrix} F^{-1} = (a, -b, -c, -d).$$

so we obtain an error when running the code. We corrected this error by taking a to have the opposite sign from b, c and d when constructing F.

There was also a minor typo in the original version of the `ImagePoint` function. In the equation on line 193, the `c` should be `cx` instead.

Finally, there were some issues with the way the conversion between Mumford and theta coordinates was done. Since we also needed to use these conversions in the glue and split cases, we simply broke out the conversion steps into separate functions and re-implemented them from scratch. We then modified the `ImagePoint` function to use theta coordinates for input and output.

3.4 Theta Coordinates and Product Varieties

In this section, we describe how to convert between points on a product of elliptic curves and theta coordinates. This conversion is necessary to complete the gluing and splitting steps of the chain.

Gluing. In this step, we are given two elliptic curves E_0 and E_A, the kernel of an (ℓ_A, ℓ_A)-isogeny

$$\Phi_{gl} \colon E_0 \times E_A \to \mathrm{Jac}(H_{gl}),$$

where H_{gl} is a hyperelliptic curve of genus 2, as well as a pair of points $\mathbf{P} = (\infty, P'_A)$ and $\mathbf{Q} = (\infty, Q'_A)$ on $E_0 \times E_A$ that we would like to map through Φ_{gl}.

The isogeny algorithm implemented in `Avisogenies` allows us to compute the codomain curve H_{gl} as well as $\Phi_{gl}(\mathbf{P})$ and $\Phi_{gl}(\mathbf{Q})$. The algorithm to recover H_{gl} takes as input the theta null point $\theta^{E_1 \times E_2}$ of $E_1 \times E_2$ along with the theta points of all the elements in the kernel of Φ_{gl}. In order to compute the image of $\mathbf{P}$, we also need to give as input the theta coordinates of $\mathbf{P}+\mathbf{K}$ for all $\mathbf{K} \in \ker \Phi_{gl}$ (and similarly for $\mathbf{Q}$).

Therefore, in order to be able to complete the gluing step, we first need to compute the theta null point of the product variety $A = E_0 \times E_A$. The first step is to compute the theta null points θ^{E_0} and θ^{E_A} of E_0 and E_A. The procedure is given in [13, §4]. However, the formulas given are for theta points of level 4, so we combine these formulas with the conversion formulas between level 2 and level 4 theta points given in [6, §3.1.2]. Starting from an elliptic curve $E : y^2 = f(x)$, to obtain the level 2 theta null point θ^E, we first compute the roots e_1, e_2 and e_3 of $f(x)$. Then, $\theta^E = (\theta_0^E : \theta_1^E)$ with

$$\theta_0^E = \sqrt{e_1 - e_3} + \sqrt{e_1 - e_2}, \qquad \theta_1^E = \sqrt{e_2 - e_3}.$$

Once we have obtained θ^{E_0} and θ^{E_A}, we can compute the theta null point of A,

$$\theta^A = (\theta^A_{00} : \theta^A_{01} : \theta^A_{10} : \theta^A_{11})$$

where

$$\theta^A_{00} = \theta^{E_0}_0 \cdot \theta^{E_A}_0, \qquad \theta^A_{01} = \theta^{E_0}_0 \cdot \theta^{E_A}_1, \qquad \theta^A_{10} = \theta^{E_0}_1 \cdot \theta^{E_A}_0, \qquad \theta^A_{11} = \theta^{E_0}_1 \cdot \theta^{E_A}_1$$

In order to compute the theta coordinates of a point $P = (R, S)$ on A we proceed in a similar way. We first compute the theta coordinates θ^R and θ^S of the elliptic curve points R and S. This conversion can be done by using the function `MumfordToLevel2ThetaPoint` that has already been implemented in the `Avisogenies` package. The theta coordinates of P are then given by

$$\theta^P_{00} = \theta^R_0 \cdot \theta^S_0, \qquad \theta^P_{01} = \theta^R_0 \cdot \theta^S_1, \qquad \theta^P_{10} = \theta^R_1 \cdot \theta^S_0, \qquad \theta^P_{11} = \theta^R_1 \cdot \theta^S_1$$

Using these two procedures, we can obtain the theta null point and all of the theta points that are needed as input to the `Avisogenies` algorithm and therefore obtain the codomain curve C and the image points $\Phi_{gl}(\mathbf{P})$ and $\Phi_{gl}(\mathbf{Q})$ completing the first step of the chain.

Splitting. In this step, we are given the Jacobian $\mathrm{Jac}(H_{sp})$ of a genus 2 hyperelliptic curve $H_{sp} : y^2 = f(x)$, as well as the kernel of an (ℓ_A, ℓ_A)-isogeny:

$$\Phi_{sp} \colon \mathrm{Jac}(H_{sp}) \to E'_0 \times C'$$

for which we know the codomain is the product of two elliptic curves E'_0 and C'. We would like to recover the images of the two divisors $D_{\mathbf{P}}$ and $D_{\mathbf{Q}}$ through Φ_{sp} where $D_{\mathbf{P}}$ and $D_{\mathbf{Q}}$ are the images of $\mathbf{P}$ and $\mathbf{Q}$ respectively through the rest of the chain. By using the functions available in `Avisogenies`, we can obtain the theta points $\theta^{\Phi(\mathbf{P})}$ and $\theta^{\Phi(\mathbf{Q})}$ of $\Phi_{sp}(D_{\mathbf{P}}) = \Phi(\mathbf{P})$ and $\Phi_{sp}(D_{\mathbf{Q}}) = \Phi(\mathbf{Q})$ along with the theta null point θ^B of $B = E'_0 \times C'$. In what follows, we will work with $\Phi(\mathbf{P})$ and write

$$\Phi(\mathbf{P}) = (P'_0, P'_C),$$

where $P'_0 \in E'_0(\mathbb{F}_{p^2})$ and $P'_C \in C'(\mathbb{F}_{p^2})$. We expect the coordinates of $\theta^{\Phi(\mathbf{P})}$ to have the form

$$\theta^{\Phi(\mathbf{P})}_{00} = \theta^{P'_0}_0 \cdot \theta^{P'_C}_0, \quad \theta^{\Phi(\mathbf{P})}_{01} = \theta^{P'_0}_0 \cdot \theta^{P'_C}_1, \quad \theta^{\Phi(\mathbf{P})}_{10} = \theta^{P'_0}_1 \cdot \theta^{P'_C}_0, \quad \theta^{\Phi(\mathbf{P})}_{11} = \theta^{P'_0}_1 \cdot \theta^{P'_C}_1.$$

If the coordinates do have this form, we can reverse the process in Section 3.4 to recover $\Phi(\mathbf{P})$. However, this case does not always occur, as theta coordinates are not unique up to isomorphism. Therefore, we might need to apply an automorphism to $\theta^{\Phi(\mathbf{P})}$ in order to transform it into the above form. The procedure to find and apply the correct isomorphism is described in the next section. For now, we assume that $\theta^{\Phi(\mathbf{P})}$ has the form above. We wish to recover P'_0, as this point is the one that should equal $\hat{\phi}_A(P'_A)$. In order to do so, the first step is to

recover $\theta_0^{P'_0}$ and $\theta_1^{P'_0}$. Since these are projective coordinates, we are really only interested in the ratio $\theta_0^{P'_0}/\theta_1^{P'_0}$. This ratio is equal to $\theta_{00}^{\Phi(\mathbf{P})}/\theta_{10}^{\Phi(\mathbf{P})}$ which we can compute directly. Once we have that, we can feed these coordinates into the `Avisogenies` function `Level2ThetaPointToMumford` to recover P'_0. This function requires as input the roots of the polynomial defining the elliptic curve on which P'_0 lies, as well as the theta null point of that curve. At first sight, there does not appear to be a problem, since we know that P'_0 must lie on E'_0 which is isomorphic to the starting curve E_0. However, we encountered the following two issues.

The first issue is that, as stated before, theta coordinates are not unique up to isomorphism, so if we give as input to `Level2ThetaPointToMumford` the theta coordinates of P'_0 and the roots of the polynomial of E_0, we get an error. Therefore, we must start by recovering the equation for E'_0 from the theta null point θ_B.

The second issue is that the points P'_0 and P'_C might have gotten swapped through the chain and so when we evaluate $\theta_{00}^{\Phi(\mathbf{P})}/\theta_{10}^{\Phi(\mathbf{P})}$ we might actually have calculated $\theta_0^{P'_C}/\theta_1^{P'_C}$ instead. In order to test whether we have the correct point, we can try running the function `Level2ThetaPointToMumford` by using the theta null point and the roots of the polynomial of the elliptic curve E'_0. If we actually gave as input the theta coordinates of P'_C instead of the ones for P'_0, we will get an error and so we can retry by using $\theta_{00}^{\Phi(\mathbf{P})}/\theta_{01}^{\Phi(\mathbf{P})}$ as $\theta_0^{P'_0}/\theta_1^{P'_0}$ instead.

Recovering the equation of E'_0 from θ^B. We assume that θ_B has the form

$$\theta_{00}^B = \theta_0^{E'_0} \cdot \theta_0^{C'}, \qquad \theta_{01}^B = \theta_0^{E'_0} \cdot \theta_1^{C'}, \qquad \theta_{10}^B = \theta_1^{E'_0} \cdot \theta_0^{C'}, \qquad \theta_{11}^B = \theta_1^{E'_0} \cdot \theta_1^{C'}.$$

If not, then we need to follow the procedure outlined in the next section to get a point of this form. We start by recovering the ratio $\theta_0^{E'_0}/\theta_1^{E'_0}$ of the theta coordinates of E'_0, by computing the ratio $\theta_{00}^B/\theta_{10}^B$. It might be the case that the curves E'_0 and C' are swapped. If so, we can detect it once we have obtained the equation for E'_0 by comparing its j-invariant with the one of E_0. If these are different, we restart the procedure by taking $\theta_{00}^B/\theta_{01}^B$ as the ratio of the theta coordinates of E'_0 instead. Recall from the gluing step that the theta coordinates of an elliptic curve $E : y^2 = f(x)$ are

$$\theta_0^E = \sqrt{e_1 - e_3} + \sqrt{e_1 - e_2}, \qquad \theta_1^E = \sqrt{e_2 - e_3}$$

We would now like to invert these equations to obtain e_1, e_2 and e_3 from θ_0^E and θ_1^E. We only care about finding one set of roots e_1, e_2, e_3 that satisfy this equation and not all of them. Therefore, we can start by setting $e_2 = 0$. Squaring the second equation, we get

$$e_3 = -(\theta_1^E)^2.$$

Substituting into the first equation and moving $\sqrt{e_1}$ to the left, we get:

$$\theta_0^E - \sqrt{e_1} = \sqrt{e_1 + (\theta_1^E)^2}.$$

Squaring both sides we get:

$$(\theta_0^E)^2 - 2\sqrt{e_1}\theta_0^E + e_1 = e_1 + (\theta_1^E)^2$$

and moving $2\sqrt{e_1}\theta_0^E$ to the right and everything else to the left, then squaring both sides, we obtain $((\theta_0^E)^2 - (\theta_1^E)^2)^2 = 4e_1(\theta_0^E)^2$ or

$$e_1 = \frac{((\theta_0^E)^2 - (\theta_1^E)^2)^2}{4(\theta_0^E)^2}.$$

Therefore, we have that the equation of an elliptic curve E with theta coordinates (θ_0^E, θ_1^E) is given by $y^2 = (x - e_1)(x - e_2)(x - e_3)$ with

$$e_1 = \frac{((\theta_0^E)^2 - (\theta_1^E)^2)^2}{4(\theta_0^E)^2}, \qquad e_2 = 0, \qquad e_3 = -(\theta_1^E)^2.$$

Once we have recovered these roots from $\theta^{E'_0}$, we have all the necessary ingredients to call the function `Level2ThetaPointToMumford` to recover P'_0. When calling this function, the order of the roots matters, so it is important to give them as a list $[e_1, e_2, e_3]$ in that order. Once we have P'_0, we can call the Magma function `IsIsomorphic` to obtain the isomorphism from E'_0 to E_0 in order to get $P_0 = \hat{\phi}_A(P'_A)$.

Applying Automorphisms to Theta Coordinates. It may be the case that after we apply the isogeny formula, we obtain a theta null point θ^B and a theta point $\theta^{\Phi(\mathbf{P})}$ that don't have a product structure, meaning that the coordinates of θ^B (and $\theta^{\Phi(\mathbf{P})}$) cannot be written as

$$\theta^B = (xu : xv : yu : yv)$$

for some elements x, y, u, v in the base field $\mathbb{F}_{p^2}$. In that case, we must apply an automorphism to θ^B before proceeding further. In order to do so, we work with the squares of the theta coordinates of level $(2, 2)$ as it is easier to detect which automorphism must be applied in that case. The squares of the theta coordinates of level $(2, 2)$ are in one-to-one correspondence with the theta coordinates of level 2. We write the (square of the) theta null point of level $(2, 2)$ of B as

$$\begin{aligned}\bar{\theta}^B = (&\bar{\theta}^B_{0000} : \bar{\theta}^B_{0001} : \bar{\theta}^B_{0010} : \bar{\theta}^B_{0011} : \bar{\theta}^B_{0100} : \bar{\theta}^B_{0101} : \bar{\theta}^B_{0110} : \bar{\theta}^B_{0111} : \\ &\bar{\theta}^B_{1000} : \bar{\theta}^B_{1001} : \bar{\theta}^B_{1010} : \bar{\theta}^B_{1011} : \bar{\theta}^B_{1100} : \bar{\theta}^B_{1101} : \bar{\theta}^B_{1110} : \bar{\theta}^B_{1111}) \in \mathbb{P}^{15}.\end{aligned}$$

We first need to convert θ^B into a point of level $(2, 2)$. The formulas for performing this conversion are given in [6, §3.1.2] and are implemented in the

`AlgebraicToAnalyticThetaNullPoint` and `AlgebraicToAnalyticThetaPoint` functions in the `Avisogenies` package. Rewriting these formulas using our notation, we have that

$$\begin{aligned}
\bar{\theta}^B_{0000} &= (\theta^B_{00}\theta^B_{00} + \theta^B_{10}\theta^B_{10} + \theta^B_{01}\theta^B_{01} + \theta^B_{11}\theta^B_{11})/4\\
\bar{\theta}^B_{0001} &= (\theta^B_{01}\theta^B_{00} + \theta^B_{11}\theta^B_{10} + \theta^B_{00}\theta^B_{01} + \theta^B_{10}\theta^B_{11})/4\\
\bar{\theta}^B_{0010} &= (\theta^B_{10}\theta^B_{00} + \theta^B_{00}\theta^B_{10} + \theta^B_{11}\theta^B_{01} + \theta^B_{01}\theta^B_{11})/4\\
\bar{\theta}^B_{0011} &= (\theta^B_{11}\theta^B_{00} + \theta^B_{01}\theta^B_{10} + \theta^B_{10}\theta^B_{01} + \theta^B_{00}\theta^B_{11})/4
\end{aligned}$$

$$\begin{aligned}
\bar{\theta}^B_{0100} &= (\theta^B_{00}\theta^B_{00} + \theta^B_{10}\theta^B_{10} - \theta^B_{01}\theta^B_{01} - \theta^B_{11}\theta^B_{11})/4\\
\bar{\theta}^B_{0101} &= (\theta^B_{01}\theta^B_{00} + \theta^B_{11}\theta^B_{10} - \theta^B_{00}\theta^B_{01} - \theta^B_{10}\theta^B_{11})/4\\
\bar{\theta}^B_{0110} &= (\theta^B_{10}\theta^B_{00} + \theta^B_{00}\theta^B_{10} - \theta^B_{11}\theta^B_{01} - \theta^B_{01}\theta^B_{11})/4\\
\bar{\theta}^B_{0111} &= (\theta^B_{11}\theta^B_{00} + \theta^B_{01}\theta^B_{10} - \theta^B_{10}\theta^B_{01} - \theta^B_{00}\theta^B_{11})/4
\end{aligned}$$

$$\begin{aligned}
\bar{\theta}^B_{1000} &= (\theta^B_{00}\theta^B_{00} - \theta^B_{10}\theta^B_{10} + \theta^B_{01}\theta^B_{01} - \theta^B_{11}\theta^B_{11})/4\\
\bar{\theta}^B_{1001} &= (\theta^B_{01}\theta^B_{00} - \theta^B_{11}\theta^B_{10} + \theta^B_{00}\theta^B_{01} - \theta^B_{10}\theta^B_{11})/4\\
\bar{\theta}^B_{1010} &= (\theta^B_{10}\theta^B_{00} - \theta^B_{00}\theta^B_{10} + \theta^B_{11}\theta^B_{01} - \theta^B_{01}\theta^B_{11})/4\\
\bar{\theta}^B_{1011} &= (\theta^B_{11}\theta^B_{00} - \theta^B_{01}\theta^B_{10} + \theta^B_{10}\theta^B_{01} - \theta^B_{00}\theta^B_{11})/4
\end{aligned}$$

$$\begin{aligned}
\bar{\theta}^B_{1100} &= (\theta^B_{00}\theta^B_{00} - \theta^B_{10}\theta^B_{10} - \theta^B_{01}\theta^B_{01} + \theta^B_{11}\theta^B_{11})/4\\
\bar{\theta}^B_{1101} &= (\theta^B_{01}\theta^B_{00} - \theta^B_{11}\theta^B_{10} - \theta^B_{00}\theta^B_{01} + \theta^B_{10}\theta^B_{11})/4\\
\bar{\theta}^B_{1110} &= (\theta^B_{10}\theta^B_{00} - \theta^B_{00}\theta^B_{10} - \theta^B_{11}\theta^B_{01} + \theta^B_{01}\theta^B_{11})/4\\
\bar{\theta}^B_{1111} &= (\theta^B_{11}\theta^B_{00} - \theta^B_{01}\theta^B_{10} - \theta^B_{10}\theta^B_{01} + \theta^B_{00}\theta^B_{11})/4.
\end{aligned}$$

Once we have obtained the coordinates of $\bar{\theta}^B$ in this way, we can separate them into two sets, the "even" coordinates:

$$\{\bar{\theta}^B_{0000}, \bar{\theta}^B_{0010}, \bar{\theta}^B_{0001}, \bar{\theta}^B_{0011}, \bar{\theta}^B_{1000}, \bar{\theta}^B_{1001}, \bar{\theta}^B_{0100}, \bar{\theta}^B_{0110}, \bar{\theta}^B_{1100}, \bar{\theta}^B_{1111}\}$$

and the "odd" coordinates:

$$\{\bar{\theta}^B_{0101}, \bar{\theta}^B_{0111}, \bar{\theta}^B_{1010}, \bar{\theta}^B_{1110}, \bar{\theta}^B_{1011}, \bar{\theta}^B_{1101}\}.$$

If we are working with a theta null point, the odd coordinates should all be zero. Furthermore, if we are working with the theta null point of a variety that is the product of two elliptic curves as we are now, exactly one of the even coordinates will be zero. In order for the corresponding theta null point of level 2 to have the desired form, we need the even zero coordinate to be $\bar{\theta}^B_{1111}$.

The process to apply an automorphism to a theta point of level $(2, 2)$ is detailed in [6, §3.1.5]. There it is only done for the case where the theta point

is defined over $\mathbb{C}$, but it is straightforward to adapt it to $\mathbb{F}_{p^2}$. An automorphism on $\bar{\theta}^B$ can be represented as a 4×4 matrix

$$\gamma = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

in the symplectic group $\mathrm{Sp}(4, \mathbb{Z})$ where A, B, C, D are 2×2 matrices with entries in $\mathbb{F}_{p^2}$. The matrix γ belongs to $\mathrm{Sp}(4, \mathbb{Z})$ if and only if the following two conditions are satisfied:

1. $A^T C$ and $D^T B$ are symmetric
2. $A^T D - C^T B = I$

where I is the 2×2 identity matrix. To write out how the matrix γ acts on the coordinates of $\bar{\theta}^B$, we first define the vector

$$d = \mathrm{diag}(A^T C) \,||\, \mathrm{diag}(D^T B) \in (\mathbb{Z}/2\mathbb{Z})^4$$

which consists of the concatenation of the diagonal entries of $A^T C$ with the diagonal entries of $D^T B$ reduced modulo 2. The matrix γ then acts in the following way on the coordinate $\bar{\theta}^B_c$:

$$\gamma \cdot \bar{\theta}^B_c = \zeta_\gamma^2 \zeta_{\gamma \cdot c}^2 \bar{\theta}^B_{c+d}$$

where ζ_γ and $\zeta_{\gamma \cdot c}$ are roots of unity in $\mathbb{F}_{p^2}$ depending on γ and γ and c respectively. These are squared because we are working with the squares of the coordinates of level $(2, 2)$. In this equation, we view the index c of the coordinate as a vector in $(\mathbb{Z}/2\mathbb{Z})^4$ which allows us to perform the addition $c+d$. Since we are working with projective coordinates and ζ_γ does not depend on the coordinate, we can treat it as a projective factor and ignore it. Therefore, it only remains to determine the value of $\zeta_{\gamma \cdot c}$. We write the index c as $c = a \,||\, b$ where a and b are vectors in $\mathbb{Z} \times \mathbb{Z}$. Note that here we look at the index as a vector over the integers and not over $\mathbb{Z}/2\mathbb{Z}$. When working over $\mathbb{C}$, the root $\zeta_{\gamma \cdot c}$ is given by

$$\zeta_{\gamma \cdot c} = \exp(-\pi i (a^T A B^T a + b^T C D^T b + 2 a^T B C^T b)).$$

We let

$$\tau = -(a^T A B^T a + b^T C D^T b + 2 a^T B C^T b)$$

and add or subtract a multiple of 2 so that $\tau \in (-1, 1]$. When computing this number in the complex plane, we have that $\zeta_{\gamma \cdot c} = e^{\pi i \tau} \in \mathbb{Q}(\sqrt{2}, i)$ whenever $\tau \in (-1, 1] \cap \frac{1}{4}\mathbb{Z}$. When working over $\mathbb{F}_{p^2}$, we can simply use the corresponding square root of 2 in $\mathbb{F}_{p^2}$ for $\sqrt{2}$ and the square root of -1 in $\mathbb{F}_{p^2}$ for i.

To choose the correct automorphism to apply, we start by identifying which of the even coordinates of $\bar{\theta}^B$ is zero. Call the index of that coordinate i. We then pick $\gamma \in \mathrm{Sp}(4, \mathbb{Z})$ so that $i = [1, 1, 1, 1] + d$ where d is defined as above and the indices are viewed as vectors in $(\mathbb{Z}/2\mathbb{Z})^4$. A suitable matrix γ for each of the

even coordinates being zero is available inside the function `get_aut_matrix` in the file `Automorphism.m`.

Once we have applied γ to $\bar{\theta}^B$, we can convert $\gamma \cdot \bar{\theta}^B$ back to level 2 by using the `Avisogenies` function `AnalyticToAlgebraicThetaPoint`. The conversion formulas are given below:

$$\begin{aligned}
\gamma \cdot \theta^B_{00} &= \gamma \cdot \theta^B_{0000} + \gamma \cdot \theta^B_{0100} + \gamma \cdot \theta^B_{1000} + \gamma \cdot \theta^B_{1100} \\
\gamma \cdot \theta^B_{01} &= \gamma \cdot \theta^B_{0001} + \gamma \cdot \theta^B_{0101} + \gamma \cdot \theta^B_{1001} + \gamma \cdot \theta^B_{1101} \\
\gamma \cdot \theta^B_{10} &= \gamma \cdot \theta^B_{0010} + \gamma \cdot \theta^B_{0110} + \gamma \cdot \theta^B_{1010} + \gamma \cdot \theta^B_{1110} \\
\gamma \cdot \theta^B_{11} &= \gamma \cdot \theta^B_{0011} + \gamma \cdot \theta^B_{0111} + \gamma \cdot \theta^B_{1011} + \gamma \cdot \theta^B_{1111}
\end{aligned}$$

The point $\gamma \cdot \theta^B = (\gamma \cdot \theta^B_{00} : \gamma \cdot \theta^B_{01} : \gamma \cdot \theta^B_{10} : \gamma \cdot \theta^B_{11})$ should now have the desired form. We then apply the same automorphism γ to the point $\theta^{\Phi(\mathbf{P})}$ so that it also has a product structure. We first need to obtain the squares of the coordinates of level $(2,2)$ of $\theta^{\Phi(\mathbf{P})}$:

$$\begin{aligned}
\bar{\theta}^{\Phi(\mathbf{P})} = (&\bar{\theta}^{\Phi(\mathbf{P})}_{0000} : \bar{\theta}^{\Phi(\mathbf{P})}_{0001} : \bar{\theta}^{\Phi(\mathbf{P})}_{0010} : \bar{\theta}^{\Phi(\mathbf{P})}_{0011} : \bar{\theta}^{\Phi(\mathbf{P})}_{0100} : \bar{\theta}^{\Phi(\mathbf{P})}_{0101} : \bar{\theta}^{\Phi(\mathbf{P})}_{0110} : \bar{\theta}^{\Phi(\mathbf{P})}_{0111} : \\
&\bar{\theta}^{\Phi(\mathbf{P})}_{1000} : \bar{\theta}^{\Phi(\mathbf{P})}_{1001} : \bar{\theta}^{\Phi(\mathbf{P})}_{1010} : \bar{\theta}^{\Phi(\mathbf{P})}_{1011} : \bar{\theta}^{\Phi(\mathbf{P})}_{1100} : \bar{\theta}^{\Phi(\mathbf{P})}_{1101} : \bar{\theta}^{\Phi(\mathbf{P})}_{1110} : \bar{\theta}^{\Phi(\mathbf{P})}_{1111}) \in \mathbb{P}^{15}.
\end{aligned}$$

These are given by

$$\begin{aligned}
\bar{\theta}^{\Phi(\mathbf{P})}_{0000} &= (\theta^B_{00}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{0001} &= (\theta^B_{01}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{0010} &= (\theta^B_{10}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{0011} &= (\theta^B_{11}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{11})/4
\end{aligned}$$

$$\begin{aligned}
\bar{\theta}^{\Phi(\mathbf{P})}_{0100} &= (\theta^B_{00}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{0101} &= (\theta^B_{01}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{0110} &= (\theta^B_{10}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{0111} &= (\theta^B_{11}\theta^{\Phi(\mathbf{P})}_{00} + \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{11})/4
\end{aligned}$$

$$\begin{aligned}
\bar{\theta}^{\Phi(\mathbf{P})}_{1000} &= (\theta^B_{00}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{1001} &= (\theta^B_{01}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{1010} &= (\theta^B_{10}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{11})/4 \\
\bar{\theta}^{\Phi(\mathbf{P})}_{1011} &= (\theta^B_{11}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{10} + \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{01} - \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{11})/4
\end{aligned}$$

$$\bar{\theta}^{\Phi(\mathbf{P})}_{1100} = (\theta^B_{00}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{11})/4$$
$$\bar{\theta}^{\Phi(\mathbf{P})}_{1101} = (\theta^B_{01}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{11})/4$$
$$\bar{\theta}^{\Phi(\mathbf{P})}_{1110} = (\theta^B_{10}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{11}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{11})/4$$
$$\bar{\theta}^{\Phi(\mathbf{P})}_{1111} = (\theta^B_{11}\theta^{\Phi(\mathbf{P})}_{00} - \theta^B_{01}\theta^{\Phi(\mathbf{P})}_{10} - \theta^B_{10}\theta^{\Phi(\mathbf{P})}_{01} + \theta^B_{00}\theta^{\Phi(\mathbf{P})}_{11})/4.$$

We then apply the automorphism γ to these coordinates by computing

$$\gamma \cdot \bar{\theta}^{\Phi(\mathbf{P})}_c = \zeta^2_\gamma \zeta^2_{\gamma\cdot c} \bar{\theta}^{\Phi(\mathbf{P})}_{c+d}$$

where d, ζ_γ and $\zeta_{\gamma\cdot c}$ are the same as above. We then convert these coordinates back to level 2 using the same formulas as before. We now have that the point

$$\gamma \cdot \theta^{\Phi(\mathbf{P})} = (\gamma \cdot \theta^{\Phi(\mathbf{P})}_{00} : \gamma \cdot \theta^{\Phi(\mathbf{P})}_{01} : \gamma \cdot \theta^{\Phi(\mathbf{P})}_{10} : \gamma \cdot \theta^{\Phi(\mathbf{P})}_{11})$$

has the form

$$\gamma \cdot \theta^{\Phi(\mathbf{P})} = (xu : xv : yu : yv).$$

4 Magma Code

The code accompanying this paper can be found at the following git repository: `https://git.uwaterloo.ca/jmlaflam/sidh-attack/`

It contains the following files:

- `Gluing.m`: This file contains implementations of the methods in Section 3.4 to convert a product of elliptic curves and points on it to theta coordinates.
- `Spitting.m`: This file contains implementations of the methods in Section 3.4 to convert theta points to points on a product of elliptic curves.
- `automorphism.m`: This file contains implementations of the methods in Section 3.4 to apply automorphisms to theta coordinates. It also contains a function `get_aut_matrix` which stores matrices for the right automorphism to apply in all possible cases.
- `ImagePoint.m`: A modified version of `Image.m` from the `Avisogenies` package which contains the modifications described in Section 3.3.
- `attack_ll.m`: This is the main file and contains an implementation of the attack when Bob's prime is $\ell_b = 5$. It contains several functions that were taken from the file `sikep751_attack.m` which can be found here.
- `parameters.txt`: This file contains the list of parameters in Table 1 along with suitable values of u, v and d which can be copy and pasted at the top of the file `attack_ll.m` to run the attack on the different parameter sets.

All of these files make use of the package `Avisogenies`, which is available at https://gitlab.inria.fr/roberdam/avisogenies/. In order to run the code, the import path for the functions coming from this package will need to be modified at the top of each file.

5 Timings and Conclusion

5.1 Timings

We ran the code in Section 4 to recover Alice's secret isogeny using different sets of parameters ℓ_A, ℓ_B, a and b. To do this, we used Magma version 2.27-8 on an Intel Core i5-7200U CPU at 2.50GHz. For some parameter sets we had to extend Alice's isogeny by a degree d isogeny so that $\ell_B^b - d\ell_A^a$ can be written as $u^2 + 4v^2$ for integers u and v. The timings of the attack given the different parameter choices are compiled in Table 1.

ℓ_A	ℓ_B	a	b	p	time
2	5	93	104	$2^{93}5^{104} - 1 \approx 2^{335}$	83 s
		216	115	$2^{216}5^{115} - 1 \approx 2^{484}$	2 mins
		109	216	$2^{109}5^{216} - 1 \approx 2^{611}$	4 mins
	7	113	106	$2^{113}7^{106} - 1 \approx 2^{411}$	38 mins
	11	99	80	$2^{99}11^{80} - 1 \approx 2^{376}$	$\approx$ 3 hours
3	5	79	88	$4 \cdot 3^{79}5^{88} - 1 \approx 2^{326}$	72 s
	7	102	73	$4 \cdot 3^{102}7^{73} - 1 \approx 2^{369}$	21 mins
5	7	107	102	$4 \cdot 5^{107}7^{102} - 1 \approx 2^{537}$	50 mins

Table 1: Timings of the attack for different choices of parameters

In [7, §5.5], the authors give a brief complexity analysis of the isogeny algorithms implemented in `Avisogenies`. The run time for the computation of each ℓ_B-isogeny depends on whether ℓ_B can be written as the sum of two squares. If it can, then the run time is in $O(\ell_B^2)$ and otherwise, ℓ_B will be written as the sum of four squares and the run time will be in $O(\ell_B^4)$. This explains the big increase in time between $\ell_B = 5$ and $\ell_B = 7$. Furthermore, when running the attack, we need to compute b-isogenies of degree ℓ_B and so the timings also depend heavily on b. Somewhat counter intuitively, the runtime of the attack when recovering Alice's isogeny does not directly depend on ℓ_A or a. However, these will influence the size of the field $\mathbb{F}_{p^2}$ and so bigger values of ℓ_A and a will make the field arithmetic slower, especially when taking square roots in the gluing step.

5.2 Future work

A natural next step is use `Avisogenies` to implement the dimension 4 and 8 attacks proposed by Robert in [17]. The isogeny formulas implemented in

the package should in theory work in higher dimensions. However, most of the implementation currently available in the package is restricted to dimension 2, specifically when it comes to the conversion to and from theta coordinates. Therefore, more work would be needed in order to extend the implementation to higher dimensions.

Our results could also be useful in implementing future attacks on M-SIDH or MD-SIDH, two SIDH variants proposed in [10] to counter the initial Castryck-Decru attack. In addition, newer isogeny-based schemes such as SQIsignHD [8], SQIsign2D-West [1], and FESTA [2] make use of isogeny interpolation in a constructive manner. Our implementation results, especially if optimized further, could lead to more flexible parameter choices or unlock other functionality.

References

1. Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQIsign2D-West: The fast, the small, and the safer. Cryptology ePrint Archive, Paper 2024/760, 2024. `https://eprint.iacr.org/2024/760`.
2. Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: Fast encryption from supersingular torsion attacks. Cryptology ePrint Archive, Paper 2023/660, 2023. `https://eprint.iacr.org/2023/660`.
3. Nils Bruin, E. Flynn, and Damiano Testa. Descent via (3,3)-isogeny on Jacobians of genus 2 curves. *Acta Arithmetica*, 165, 01 2014.
4. David G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48(177):95–101, 1987.
5. Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 423–447, Cham, 2023. Springer Nature Switzerland.
6. Romain Cosset. *Applications des fonctions thêta à la cryptographie sur courbes hyperelliptiques.* Theses, Université Henri Poincaré - Nancy I, November 2011.
7. Romain Cosset and Damien Robert. Computing (ℓ, ℓ)-isogenies in polynomial time on jacobians of genus 2 curves. *Mathematics of Computation*, 84(294):1953–1975, 2015.
8. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. Sqisignhd: New dimensions in cryptography. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology — EUROCRYPT 2024*, pages 3–32, Cham, 2024. Springer Nature Switzerland.
9. Thomas Decru and Sabrina Kunzweiler. Efficient computation of $(3^n, 3^n)$-isogenies. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023*, pages 53–78, Cham, 2023. Springer Nature Switzerland.
10. Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. M-SIDH and MD-SIDH: Countering SIDH attacks by masking information. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 282–309, Cham, 2023. Springer Nature Switzerland.
11. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 19–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

12. Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik*, 1997(485):93–122, 1997.
13. Markus Kirschmer, Fabien Narbonne, Christophe Ritzenthaler, and Damien Robert. Spanning the isogeny class of a power of an elliptic curve. *Math. Comp.*, 91(333):401–449, 2021.
14. KULeuven-COSIC. (3,3)-isogenies. `https://github.com/KULeuven-COSIC/3_3_isogenies`, 2023.
15. Rémy Oudompheng and Giacomo Pope. A note on reimplementing the Castryck-Decru attack and lessons learned for sagemath. Cryptology ePrint Archive, paper 2022/1283, 2022. `https://eprint.iacr.org/2022/1283`.
16. Damien Robert. avisogenies magma package, version 0.7. `https://gitlab.inria.fr/roberdam/avisogenies/`, 2021.
17. Damien Robert. Breaking sidh in polynomial time. In *Advances in Cryptology — EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, pages 472–503, Berlin, Heidelberg, 2023. Springer-Verlag.
18. Damien Robert. On the efficient representation of isogenies. Cryptology ePrint Archive, paper 2024/1071, 2024. `https://eprint.iacr.org/2024/1071`.
19. Benjamin Andrew Smith. *Explicit endomorphisms and correspondences*. PhD thesis, University of Sydney, 2005-12-23.
20. J. Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences, Série I*, 273:238–241, juillet 1971.