Implementing the Castryck-Decru attack on SIDH with general primes

by

Jeanne Laflamme

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With the rapid progress of quantum computers in recent years, efforts have been made to standardize new public-key cryptographic protocols which would be secure against them. One of the schemes in contention was Supersingular Isogeny Diffie-Hellman (SIDH). This scheme relied on the assumed hardness of the isogeny problem on supersingular elliptic curves. However, in the SIDH protocol extra information on the secret isogenies is transmitted. In July 2022, Castryck and Decru found a way to exploit this information to completely break the scheme. They gave an implementation of their attack which allows to recover Bob's secret key in a few seconds on a laptop. Usually, Alice and Bob's secret isogenies are taken to have degree 2^a and 3^b respectively. This thesis gives a more general implementation of the attack in Magma which works even if Alice and Bob's secret isogenies have degrees ℓ_A^a and ℓ_B^b for more general primes ℓ_A and ℓ_B.

Acknowledgements

I would like to thank my supervisor David Jao for all of his time and help. I would also like to thank Damien Robert for answering my questions on `Avisogenies` and theta coordinates. I am also very grateful to all of my friends who made my time in Waterloo enjoyable. Finally, I would like to thank my mom and my cousins for their precious support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Public-Key Cryptography

Public-key cryptography gives a way for two parties to establish a shared secret key over an insecure channel. The first such key-exchange algorithms to be commercialized include RSA and Elliptic Curve Diffie-Hellman (ECDH) which were developed in 1977 and 1976 respectively. These remain the most used public-key cryptosystems today. Public-key cryptosystems rely on the use of hard mathematical problems. There are said to be asymmetric because, although the underlying mathematical problem is difficult to solve (i.e. it would be infeasible to solve in a reasonable amount of time with our current computing capabilities), checking the correctness of a solution is easy. For example, RSA relies on the difficulty of factoring large integers and ECDH relies on the difficulty of solving discrete logarithms on elliptic curves.

While no serious classical threats have yet been found for RSA or ECDH, in 1994, Shor [13] discovered a polynomial-time quantum algorithm which can be applied to solving both integer factorization and discrete logarithms. At the time, quantum computers were a purely theoretical concept, but in recent years, significant investments have been put in to build them. While we are still ways away from having quantum computers powerful enough to factor RSA-sized integers, we need to prepare for this eventuality. This prompted the National Institute for Standards and Technology (NIST) to launch a competition for the standardization of new public-key cryptosystems that are believed to be secure against quantum computers.

1.2 Isogeny-based Cryptography

One of the schemes proposed for the NIST competition was Supersingular Isogeny Diffie-Hellman (SIDH). This key-exchange protocol was first proposed by Jao and De Feo in 2011 [8]. This scheme relied on the assumed hardness of the isogeny problem on supersingular elliptic curves which will be introduced below. In July 2022, SIDH advanced to the fourth and final round of the NIST competition as an alternate candidate. However, a few weeks later, Castryck and Decru [3] discovered an attack that allowed to recover Bob's secret isogeny in a few hours on a laptop. Later improvements have made it possible to run the attack in a few seconds [11].

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 to compute a degree 2^a isogeny in dimension 2. Formulas by Richelot [15] allow to do this efficiently and make the attack possible. If instead, we wanted to recover Alice's secret isogeny, we would need to compute a degree 3^b isogeny in dimension 2. Formulas to do this have also been developed in [2] and optimized in [6]. An implementation of the attack that works to recover Alice's isogeny has been done in Magma.

This prompts the question of whether the attack would still be feasible if Alice and Bob's secret isogenies had degrees ℓ_A^a and ℓ_B^b where ℓ_A and ℓ_B are primes larger than 2 and 3. Implementing the attack in that case would require to compute a degree ℓ^e isogeny in dimension 2 for $\ell > 3$. Formulas to do this have been developed by Cosset and Robert [5]. They have been mostly implemented in a Magma package called `Avisogenies`. However, the package lacked implementations for some of the necessary steps of the attack, namely the "glue" and "split" cases which are explained below. The main contribution of this thesis is to implement these and to use them to give a working implementation for the Castrick-Decru attack when Alice and Bob use general primes ℓ_A and ℓ_B in the SIDH protocol.

Chapter 2

Background on Elliptic Curves and Abelian Varieties

In this chapter, we will present the background theory on elliptic curves that is necessary to understand the SIDH protocol. We will also introduce some background information on abelian varieties of genus 2 as the Castryck-Decru attack relies on these higher dimensional varieties.

2.1 Elliptic Curves

Definition 2.1.1 (Elliptic Curve)**.** *An* ***elliptic curve*** E *is the set of solutions to an equation of the form*

$$y^2 = x^3 + Ax + B,$$

where the discriminant of the polynomial $f(x) = x^3 + Ax + B$ *is non zero. If* K *is a field with* $A, B \in K$*, we say that* E *is defined over* K *and write* E/K*.*

We can look at the set of points (x, y) on E such that $x, y \in L$ for any field $L \supseteq K$. We denote this set by $E(L)$. In this thesis, we will mostly consider elliptic curves defined over a finite field $\mathbb{F}_{p^2}$ where p is a prime number.

The most interesting feature of elliptic curves is that the set of points $E(L)$ forms an abelian group. However, in order to define a group law that satisfies the four axioms of a group, we need to add an extra point to $E(L)$ which is called the *point at infinity* and is usually denoted by ∞ or $\mathcal{O}$. This point will serve as the identity element.

Group Law

Let E be an elliptic curve and $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points on E. The group law is defined as follows:

1. If $x_1 \neq x_2$ then trace the line going through P_1 and P_2. Let (x_3, y_3) denote the coordinates of the third point of intersection of this line with the curve. Then, $P_1 + P_2 = (x_3, -y_3)$
2. If $x_1 = x_2$ and $y_1 = -y_2$ then $P_1 + P_2 = \infty$ (i.e. $P_1 = -P_2$)
3. If $P_1 = P_2$ and $y_1 \neq 0$ then trace the line tangent to the curve at P_1 and proceed as in 1. (point doubling)
4. $P + \infty = P$ for all P on E. (∞ is the identity)

The commutativity, identity and inverses properties of a group are easily verified from this definition. It is straightforward to translate these geometric manipulations into algebraic formulas that can be used to quickly compute the sum of any two points on the curve. We can then use these formulas to check that $P_1 + (P_2 + P_3) = (P_1 + P_2) + P_3$ for any three points on the curve, which then proves associativity. Computer packages such as Sage or Magma have efficient implementations of elliptic curve arithmetic.

Let E be an elliptic curve defined over $\mathbb{F}_{p^2}$. We can look at the set of points on E over $\overline{F}_{p^2}$ that have order dividing n for any integer n. We denote this set $E[n]$:

Definition 2.1.2 (n-torsion Subgroup)**.** *The* ***n-torsion subgroup*** *of $E/\mathbb{F}_{p^2}$ is the set of points in $E(\overline{F}_{p^2})$ whose order divides n:*

$$E[n] = \{P \in E(\overline{F}_{p^2}) \quad | \quad nP = \infty\}.$$

It can be shown that if p doesn't divide n then $E[n] \simeq \mathbb{Z}_n \times \mathbb{Z}_n$. The group $E[p]$ will be either $\{\infty\}$ or isomorphic to $\mathbb{Z}_p$. This allows us to make the following definition:

Definition 2.1.3 (Supersingular Elliptic Curve)**.** *An elliptic curve E defined over $\mathbb{F}_{p^2}$ is* ***supersingular*** *if $E[p] = \{\infty\}$. That is, if it has no points of order p.*

The main interest of supersingular elliptic curves is that we have $E(\mathbb{F}_{p^2}) \simeq \mathbb{Z}_{p+1} \times \mathbb{Z}_{p+1}$ for any supersingular elliptic curve E and prime p. Therefore, if we work with a prime p that has the form $p = AB - 1$, any supersingular elliptic curve will have its full A and B torsion defined over $\mathbb{F}_{p^2}$.

2.2 Isogenies

Definition 2.2.1 (Isogeny)**.** *Let E_1 and E_2 be elliptic curves. An **isogeny** between E_1 and E_2 is a rational map*

$$\phi : E_1 \to E_2$$

satisfying $\phi(\infty) = \infty$.

It turns out that only requiring $\phi(\infty) = \infty$ implies that $\phi(P + Q) = \phi(P) + \phi(Q)$ for any points P and Q. The proof of this can be found in theorem 4.8 of chapter 3 in [14]. Therefore, any isogeny is a homomorphism. Since ϕ is a rational map, this means there exist rational functions $\phi_1(x, y)$ and $\phi_2(x, y)$ such that

$$\phi(x, y) = (\phi_1(x, y), \phi_2(x, y)).$$

Using the relation $y^2 = x^3 + Ax + B$ and the fact that ϕ is a homomorphism, we can write

$$\phi_1(x, y) = \frac{p_1(x)}{q_1(x)}$$

and

$$\phi_2(x, y) = \frac{p_2(x)}{q_2(x)}y.$$

Here, p_1, q_1, p_2 and q_2 are polynomials in $\mathbb{F}_{p^2}[x]$. The exact procedure to reduce isogenies to this form can be found in section 2.9 of [17].

This allows us to give the following simple definition of the degree of an isogeny:

Definition 2.2.2 (degree of an isogeny)**.** *Let E_1 and E_2 be elliptic curves defined over $\mathbb{F}_{p^2}$ and $\phi : E_1 \to E_2$ be an isogeny between the two. The **degree** of ϕ is*

$$\deg(\phi) = \max\{\deg(p_1), \deg(q_1)\}.$$

An alternative and more standard definition for the degree of an isogeny can be found in [14] section 3.4. An isogeny of degree one is an isomorphism. The following definition will be useful in determining which elliptic curves are isomorphic to one another

Definition 2.2.3 (j-invariant)**.** *Let $E : y^2 = x^3 + ax + b$ be an elliptic curve defined over $\mathbb{F}_{p^2}$. The **j-invariant of E** is*

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2}.$$

If E and E' are two elliptic curves defined over $\mathbb{F}_{p^2}$, then $E(\overline{F}_{p^2}) \simeq E'(\overline{F}_{p^2})$ if and only if $j(E) = j(E')$.

Going back to general-degree isogenies, a simple example that works on any curve is the multiplication by m map:

Example 2.2.1. *Let E be an elliptic curve. The multiplication by m map*

$$\begin{aligned}[m] : E &\to E \\ P &\mapsto mP\end{aligned}$$

is an isogeny.

Other useful isogenies are the Frobenius morphisms:

Example 2.2.2. *If E is defined over a field of characteristic p and $q = p^r$ for some integer r, then we can define the isogeny*

$$\phi_q : E \to E^{(q)}$$

as

$$\phi_q(x, y) = (x^q, y^q)$$

*This isogeny is called the q^{th}-power **Frobenius morphism**.*

In what follows, we will be concerned with computing isogenies that are "separable". Separable isogenies are nice to work with as their codomain is fully determined by their kernel. Furthermore, we have that if ϕ is separable then $\deg(\phi) = \#\ker(\phi)$. For a formal definition of separable isogenies and a proof of these facts see [14] section III.4. Every isogeny can be written as the composition of a frobenius morphism with a separable isogeny (see [14] II.2.12 for a proof). Once we know the kernel of an isoeny, we can compute its codomain and its image on any point efficiently by using Velu's formulas [16]. Isogenies do not have inverses. However, we can work with the following:

Proposition 2.2.1 (Dual Isogeny)**.** *Let E_1 and E_2 be elliptic curves and $\phi : E_1 \to E_2$ be an isogeny between them with $\deg \phi = m$. Then, there exists a unique isogeny*

$$\hat{\phi} : E_2 \to E_1$$

*such that $\hat{\phi} \circ \phi = [m]$. This isogeny is called the **dual** of ϕ.*

Proof. See [14] III.6.1a. □

2.3 Divisors

The Castryck-Decru attack on SIDH relies on the computation of isogenies on genus 2 abelian varieties. In order to generalize the preceding concepts to higher genus, we first need to introduce the notion of a divisor. For simplicity, we begin by defining divisors of elliptic curves.

Definition 2.3.1 (Divisor of a cuvre)**.** *Let $E/\mathbb{F}_{p^2}$ be an elliptic curve. The* ***set of divisors*** *of E, denoted Div(E) is the free abelian group generated by the formal symbols (P) for $P \in E(\overline{F}_{p^2})$. In other words, a divisor is a formal sum:*

$$D = \sum_j a_j(P_j),$$

where $a_j \in \mathbb{Z}$ and $P_j \in E(\overline{F}_{p^2})$.

We can then introduce the notion of degree of a divisor:

Definition 2.3.2 (Degree of a divisor)**.** *The* ***degree*** *of a divisor $D = \sum_j a_j(P_j)$, denoted* $\deg(D)$ *is*

$$\deg(D) = \sum_j a_j.$$

We will be particularly interested in divisors of degree 0. The set of such divisors is denoted $\mathrm{Div}^0(E)$. Another set of divisors that will be of interest are principal divisors. To define these, we need to first introduce the concept of functions on an elliptic curve.

Definition 2.3.3 (Function on an Curve)**.** *A function on an elliptic curve $E/\mathbb{F}_{p^2}$ is a rational function $f(x, y) \in \overline{F}_{p^2}(x, y)$ which takes as input points on E and returns an element of $\overline{F}_{p^2}$ or ∞.*

$$f : U \to \overline{F}_{p^2} \cup \{\infty\}$$

where $U \subseteq E(\overline{F}_{p^2})$.

By following a similar procedure as with isogenies, we can always write f as

$$f(x, y) = \frac{p(x)}{q(x)}y,$$

where $p, q \in \overline{F}_{p^2}[x]$. If $f(P) = 0$, we say that f has a **zero** at P and if $f(P) = \infty$ (i.e. $q(P) = 0$), we say that f has a pole at P. For a fixed point P, we can always rewrite f as

$$f = u_P^r g,$$

where $g(P) \neq 0$ or ∞ and u_P is a rational function in $\overline{F}_{p^2}(x, y)$ such that $u_P(P) = 0$. We say that r is the **order** of f at P and denote this by $\text{ord}_P(f)$. This allows us to introduce another type of divisors

Definition 2.3.4 (Divisor of a function)**.** *Let $E/\mathbb{F}_{p^2}$ be an elliptic curve and f be a function on E. The **divisor of f** is defined as*

$$div(f) = \sum_P ord_P(f)(P).$$

Since we are working over finite fields, the sum above is finite and so $\text{div}(f) \in Div(E)$.

Proposition 2.3.1. *Let $E/\mathbb{F}_{p^2}$ be an elliptic curve and f be a function on E. Then,*

- $\deg(div(f)) = 0$
- *If $div(f) = 0$ then f is constant.*

Proof. See [14] Section 2.3, proposition 3.1 □

Definition 2.3.5 (Principal divisor)**.** *A divisor $D \in Div(E)$ is **principal** if it can be written as $D = div(f)$ for some function f on E.*

The use of principal divisors is to define an equivalence relation on elements of $\text{Div}^0(E)$.

Definition 2.3.6 (Divisor equivalence)**.** *Let D_1 and D_2 be divisors on a elliptic curve E. We say that D_1 is **linearly equivalent** to D_2 and write $D_1 \sim D_2$ if $D_1 - D_2$ is principal.*

This equivalence relation is the building block for two very important groups:

Definition 2.3.7 (Divisor class group)**.** *Let E be an Elliptic Curve. The **Divisor class group** of E, also called the **Picard group**, is*

$$Pic(E) = Div(E)/\sim .$$

It follows from proposition 2.3.1 that the set of principal divisors is a subset of $\mathrm{Div}^0(E)$. Therefore, we can also define the following group:

Definition 2.3.8 (Jacobian)**.** *Let E be an elliptic curve. The **Jacobian** of E, denoted $Jac(E)$ or $Pic^0(E)$ is*

$$Jac(E) = Div^0(E)/\sim .$$

It is actually the case that for elliptic curves, $\mathrm{Jac}(E) \simeq E(\overline{F}_{p^2})$. The simplest way to go from divisors to points on the curve is through the following natural map:

$$\begin{aligned} \nu : \mathrm{Div}(E) &\longrightarrow E(\overline{F}_{p^2}) \\ \sum_j a_j(P_j) &\mapsto \sum_j a_j P_j. \end{aligned}$$

We will be interested in the restriction of this map to the set of divisors of degree 0, $\mathrm{Div}^0(E)$. Clearly, this map is surjective since

$$\nu((P) - (\infty)) = P$$

for any point P on E. Furthermore, it can be shown that the kernel of ν is the set of principal divisors. A proof of this can be found in [17] section 11.1, theorem 11.2. Therefore, it follows from the first isomorphism theorem that

$$\mathrm{Jac}(E) \simeq E(\overline{F}_{p^2}).$$

We can deduce from this isomorphism that every element of $\mathrm{Jac}(E)$ has a unique representative of the form $(P) - (\infty)$ for some $P \in E(\overline{F}_{p^2})$.

2.4 Abelian Varieties of Genus 2

Elliptic curves are abelian varieties of genus one. We will now introduce another type of abelian varieties that are a generalization of elliptic curves to higher genera:

Definition 2.4.1 (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 eliminate the h. Since we will be working with fields $\mathbb{F}_{p^2}$ for large primes p, this will always be the case and so in the future, we will 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 we did above for elliptic curves and work with the group $\mathrm{Jac}(C)$ instead. 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***.

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

2.5 Isogenies in Dimension 2

We can look at isogenies between abelian varieties in dimension 2. Specifically, we will be interested in isogenies

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

that go from a product of two elliptic curves to the product of two elliptic curves. Such isogenies were characterized by Kani's theorem [9]. They can be written in matrix form

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

where $\alpha : E_1 \to E_3$, $\beta : E_2 \to E_3$, $\gamma : E_1 \to E_4$ and $\delta : E_2 \to E_4$ are dimension one isogenies of elliptic curves. We will be working with separable isogenies and so the number of points in $\ker \Phi$ determines its degree d. 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 Velu's formulas that allow us to compute the codomain of dimension 2 isogenies from their kernel.

By factoring d, we can break up Φ into the 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 $\mathbb{F}_{p^2}$, 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 will first need to map it through a "Glue" isogeny

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

then we map it through $e - 2$ isogenies between Jacobians

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

and finally, we map it through a "split" isogeny:

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

Formulas due to Richelot [15] address the three cases when $\ell = 2$. When $\ell = 3$, explicit formulas have also been found [2, 6]. In the case of a generic prime ℓ, Cosset and Robert [5] have developed formulas that make use of theta coordinates. These are the formulas we will be working with.

2.6 Mumford Coordinates

Mumford coordinates give a practical way to represent divisors on the Jacobian of hyperelliptic curves. As stated before, in the case of genus 2, any element in the Jacobian can be represented uniquely as a reduced divisor:

$$D = (P) + (Q) - 2(\infty).$$

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 gives a way to efficiently add divisors when they are written in this form.

Chapter 3

Supersingular Isogeny Diffie-Hellman

Supersingular Isogeny Diffie-Hellman 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 [8]. It was submitted to the United States National Institute of Standards and Technology's (NIST) competition for standardization of post-quantum cryptography protocols. It advanced to the fourth round of the competition as an alternate candidate. However, in July 2022, Castryck and Decru [3] 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 Odompheng [11], made it possible for the attack to run in a few seconds.

3.1 Protocol

To start the protocol, Alice and Bob must 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 will be considering the case where larger values of ℓ_A and ℓ_B are chosen. From here, Alice and Bob pick a starting supersingular elliptic curve E_0 which is defined over $\mathbb{F}_{p^2}$. This is usually taken to be $E_0 : y^2 = x^3 + 6x^2 + x$. Alice and Bob can then 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 will both start by choosing a secret integer sk_A and sk_B. Alice will then compute the codomain E_A of the ℓ_A^a-isogeny ϕ_A whose kernel is generated by $P_A + sk_A Q_A$. She will then transmit 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 will do

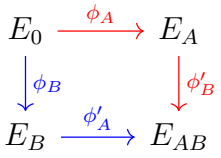

Figure 3.1: SIDH key exchange diagram

the corresponding thing: he will choose a secret integer sk_B and construct an isogeny $\phi_B : E_0 \to E_B$ whose kernel is $P_B + sk_B Q_B$. He will then transmit $E_B, \phi_B(P_A)$ and $\phi_B(Q_A)$ to Alice. With this information, Alice will compute the codomain E_{AB} of the isogeny $\phi'_A : E_B \to E_{BA}$ whose kernel is generated by $\phi_B(P_A) + sk_A\phi_B(Q_A)$. Bob will do the corresponding computation by calculating the codomain E_{AB} of the isogeny $\phi'_B : E_A \to E_{AB}$ whose kernel is generated by $\phi_A(P_B) + sk_B\phi_A(Q_B)$. Alice and Bob will land on isomorphic curves $E_{AB} \simeq E_{BA}$ and so they can use the j-invariant of these curves as their shared secret key. The key exchange protocol is summarized in figure 3.1.

The security of this protocol was based on the assumed hardness of the isogeny problem. This hardness assumption states that, given two isogenous elliptic curves E_1 and E_2, it is computationally infeasible to recover 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 believed for a long time that the transmission of this extra information did not impact the security of the scheme. However, Castryck and Decru [3] showed how it could be exploited to completely break the scheme.

3.2 Castryck-Decru Attack

In what follows, we will be working to recover Alice's isogeny ϕ_A. Since we are working with arbitrary ℓ_A and ℓ_B, this 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. These chains were characterized by Kani [9]. The goal is to construct such an isogeny chain for which the kernel can be calculated from the torsion point information and where one of the components gives us the image of $\hat{\phi}_A$ on any point of E_A, when mapping a suitable point through the chain. Once this is done, we can compute $\hat{\phi}_A$ on generators P'_A and Q'_A of the ℓ_A^a-torsion on E_A. We then know that the kernel of ϕ_A is generated by either $\hat{\phi}_A(P'_A)$ or $\hat{\phi}_A(Q'_A)$. Castryck and Decru originally devised this attack in dimension 2 but Robert [12] later generalized their idea to dimensions

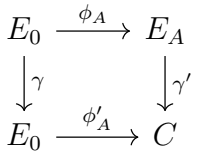

Figure 3.2: Isogeny Square used in the Castryck-Decru attack

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 first 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 and so the chain indeed splits at the end. We can then build the isogeny square in figure 3.2.

The dimension 2 isogeny chain will be

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

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 3.1. If γ can be computed efficiently, this kernel can be obtained almost 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

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

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 will have 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 [6].

Chapter 4

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 dimension 2 (ℓ, ℓ)-isogenies that starts with a gluing step and ends with a splitting step for an arbitrary prime ℓ. Cosset and Robert [5] have devised algorithms to compute (ℓ, ℓ)-isogenies between abelian varieties of genus g that make use of theta coordinates. These algorithms are implemented in a Magma package called `Avisogenies`.

4.1 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 4.1.1 (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 genus g in the projective plane $\mathbb{P}^{n^g-1}$. This works regardless of which type of abelian variety we are working with.

Therefore, in genus 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

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

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 the projective plane, we will use a basis for that vector space. Different choices of bases will 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 will be mainly working with theta functions of level 2 and genus 2 and therefore we will be working on $\mathbb{P}^3$.

The original Riemann theta functions were defined over the complex numbers. Abelian varieties of genus 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 $\mathcal{H}_g$. Here however, 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 characteristic 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 will 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 $\Lambda_\Omega = \Omega\mathbb{Z}^2 + \mathbb{Z}^2$. To do this, we first need to 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 plane 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 [4] 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 [5] and are implemented in `Avisogenies`. We will 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 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 will abbreviate the coordinate $\theta_{ij}(0,\Omega)$ as θ_{ij}^A.

In section 4.4.3, we will 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:

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

This is sometimes referred to as level $(2,2)$ because of the choice of basis. These functions are sometimes renumbered using indices 1 to 16. Different numberings were used by different people. To avoid confusion, we will stick to labelling the coordinates using the four binary digits. Some of the numberings used by other authors are explicited in [4] section 3.1.2.

4.2 The `Avisogenies` Package

`Avisogenies` is a Magma package developed by Damien Robert and Romain Cosset. It contains implementations of algorithms to compute genus 2 (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 can be used for any genus 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`. They are so called because they 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 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 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, the following things needed to be done:

- Implement the conversion from an elliptic curve to its theta null point
- Implement the conversion from the theta null point of two elliptic curves E_1 and E_2 to the theta null point of the product $E_1 \times E_2$
- Implement the conversion from a point on a product of elliptic curves to theta coordinates
- Fix the `ImagePoint` function and reorganize it so that it takes as input and outputs theta coordinates and so can be also used for the glue and split cases
- Implement the conversion from a theta null point of the product of two elliptic curves to the theta null point of each elliptic curve.
- Implement the conversion from a theta null point to an elliptic curve
- Implement the conversion from the theta coordinates of a point (P, Q) on the product of two elliptic curves to the points P and Q.

4.3 Computing the Image of a Point

Algorithm 4.5 in [5] 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 had not been tested 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

$$F = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

or

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

We will deal with the 4×4 case but the same applies to the 2×2 case. In step 4, we need to take 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. This suggests that we should take a to have the opposite sign from b, c and d when constructing F. Indeed, taking the negative root of a^2 instead of the positive one solved the problem.

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. However we needed to be able to use this function in the glue and split cases too and so we simply removed this from the function so that it would take as input and output theta coordinates. The conversions for the three cases (glue, split and Jacobian to Jacobian) were done from scratch in separate functions.

4.4 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 is necessary to complete the gluing and splitting steps of the chain.

4.4.1 Gluing

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

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

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 [10] section 4. However, the formulas given are for theta points of level 4 and so we combine these with the conversion formulas between level 2 and level 4 theta points which are given in [4] section 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, $\theta^E = (\theta^E_0 : \theta^E_1)$ with

$$\begin{aligned} \theta^E_0 &= \sqrt{e_1 - e_3} + \sqrt{e_1 - e_2} \\ \theta^E_1 &= \sqrt{e_2 - e_3}. \end{aligned}$$

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

$$\theta_{00}^{A} = \theta_{0}^{E_0} \cdot \theta_{0}^{E_A}$$
$$\theta_{01}^{A} = \theta_{0}^{E_0} \cdot \theta_{1}^{E_A}$$
$$\theta_{10}^{A} = \theta_{1}^{E_0} \cdot \theta_{0}^{E_A}$$
$$\theta_{11}^{A} = \theta_{1}^{E_0} \cdot \theta_{1}^{E_A}$$

In order to compute the theta coordinates of a point $P = (R, S)$ on A we proceed in a very 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 will then be given by

$$\theta_{00}^{P} = \theta_{0}^{R} \cdot \theta_{0}^{S}$$
$$\theta_{01}^{P} = \theta_{0}^{R} \cdot \theta_{1}^{S}$$
$$\theta_{10}^{P} = \theta_{1}^{R} \cdot \theta_{0}^{S}$$
$$\theta_{11}^{P} = \theta_{1}^{R} \cdot \theta_{1}^{S}$$

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.

4.4.2 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:

$$\Phi_{sp} : \mathrm{Jac}(H_{sp}) \to E_0' \times C'$$

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

$$\begin{aligned}
\theta_{00}^{\Phi(\mathbf{P})} &= \theta_0^{P'_0} \cdot \theta_0^{P'_C} \\
\theta_{01}^{\Phi(\mathbf{P})} &= \theta_0^{P'_0} \cdot \theta_1^{P'_C} \\
\theta_{10}^{\Phi(\mathbf{P})} &= \theta_1^{P'_0} \cdot \theta_0^{P'_C} \\
\theta_{11}^{\Phi(\mathbf{P})} &= \theta_1^{P'_0} \cdot \theta_1^{P'_C}.
\end{aligned}$$

If this is the case, we can reverse the process in section 4.4.1 to recover $\Phi(\mathbf{P})$. However, this does not always happen as theta coordinates are not unique up to isomorphism. Therefore, we might need to apply an automorphism to $\theta^{\Phi(\mathbf{P})}$ so that it has the form above. The procedure to find and apply the correct isomorphism is described in the next section. For now, we will assume that $\theta^{\Phi(\mathbf{P})}$ has the form above. We will be working to recover P'_0 as this is the point that should equal $\hat{\phi}_A(P'_A)$. In order to do this, 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 also takes 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. This 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, there are two issues that we run into.

The first one is that, as stated before, theta coordinates are not unique up to isomorphism and 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 will assume for now that θ_B has the form

$$\begin{aligned}\theta^B_{00} &= \theta_0^{E'_0} \cdot \theta_0^{C'} \\ \theta^B_{01} &= \theta_0^{E'_0} \cdot \theta_1^{C'} \\ \theta^B_{10} &= \theta_1^{E'_0} \cdot \theta_0^{C'} \\ \theta^B_{11} &= \theta_1^{E'_0} \cdot \theta_1^{C'}.\end{aligned}$$

If that is not the case then we need to start by following the procedure outlined in section 4.4.3 to get a point of this form. We start by recovering the ratio of the theta coordinates of E'_0 $\theta_0^{E'_0}/\theta_1^{E'_0}$ by taking it to be the ratio $\theta^B_{00}/\theta^B_{10}$. It might be the case that the curves E'_0 and C' are swapped. If this happens, 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 $\theta^B_{00}/\theta^B_{01}$ as the ratio of the theta coordinates of E'_0 instead. In the gluing step, we explained how to obtain the theta coordinates of an elliptic curve $E : y^2 = f(x)$ from the roots e_1, e_2, e_3 of f. We simply took

$$\begin{aligned}\theta_0^E &= \sqrt{e_1 - e_3} + \sqrt{e_1 - e_2} \\ \theta_1^E &= \sqrt{e_2 - e_3}\end{aligned}$$

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 satidfy this equation and not all of them. Therefore, we can start by setting $e_2 = 0$. Squaring the second equation, we get

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

Replacing these values in 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:

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

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

$$\begin{aligned} e_1 &= \frac{((\theta_0^E)^2 - (\theta_1^E)^2)^2}{4(\theta_0^E)^2}. \\ e_2 &= 0 \\ e_3 &= -(\theta_1^E)^2. \end{aligned}$$

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 matter 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 $P_0 = \hat{\phi}_A(P'_A)$.

4.4.3 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. By that we mean 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 this, we will 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 will 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 conversion formulas are given in section 3.1.2 of [4] and are implemented in the functions `AlgebraicToAnalyticThetaNullPoint` and `AlgebraicToAnalyticThetaPoint` 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 [4] section 3.1.5. 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 will then act in the following way on the coordinate $\bar{\theta}^B_c$:

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

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

$$e = -(a^T AB^T a + b^T CD^T b + 2a^T BC^T b)$$

and add or substract a multiple of 2 so that $e \in (-1, 1]$. When writing this root in cartesian coordinates, we have that:

$$\begin{aligned}
\text{When } e = -\frac{3}{4} &: \zeta_{\gamma\cdot c} = -\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i \\
\text{When } e = -\frac{1}{2} &: \zeta_{\gamma\cdot c} = -i \\
\text{When } e = \frac{1}{4} &: \zeta_{\gamma\cdot c} = \frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}i \\
\text{When } e = 0 &: \zeta_{\gamma\cdot c} = 1 \\
\text{When } e = \frac{1}{4} &: \zeta_{\gamma\cdot c} = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i \\
\text{When } e = \frac{1}{2} &: \zeta_{\gamma\cdot c} = i \\
\text{When } e = \frac{3}{4} &: \zeta_{\gamma\cdot c} = -\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i \\
\text{When } e = 1 &: \zeta_{\gamma\cdot c} = -1
\end{aligned}$$

When working over $\mathbb{F}_{p^2}$, we can simply use the corresponding root where $\sqrt{2}$ is a square root of 2 in $\mathbb{F}_{p^2}$ and i is a square root of -1 in $\mathbb{F}_{p^2}$.

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 gamma for each of the even coordinate being zero is recorded inside the function `get_aut_matrix` in the file `Automorphism.m` (see appendix A).

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 will then need to 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 \\
\\
\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 \\
\\
\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 \\
\\
\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.
\end{aligned}$$

We then apply the automorphism γ to these coordinates by computing

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

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 should now have that the point

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

has the form

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

Chapter 5

Timings and Conclusion

5.1 Timings

We ran the code in appendix A 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 5.1.

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 5.1: Timings of the attack for different choices of parameters

In [5] section 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 we need to take square roots in the gluing step.

5.2 Future work

A natural next step would be to use `Avisogenies` to implement the dimension 4 and 8 attacks proposed by Robert in [12]. The isogeny formulas implemented in the package should in theory work in higher dimension. However, most of the implementation currently available in the package is restricted to genus 2, specifically when it comes to the conversion to and from theta coordinates. Therefore, more implementation work needs to be done before this is possible.

This work could also be useful in implementing future attacks on M-SIDH or MD-SIDH, two SIDH variants proposed in [7] to counter the initial Castryck-Decru attack. Finally, FESTA, another isogeny-based scheme proposed in [1] makes use of the Castryck-Decru attack in its key exchange protocol. Our current (ℓ, ℓ)-isogeny chain could allow for more flexible parameter choices in this scheme, provided that it was optimized sufficiently.

References

[1] 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.

[2] Nils Bruin, E. Flynn, and Damiano Testa. Descent via (3,3)-isogeny on jacobians of genus 2 curves. *Acta Arithmetica*, 165, 01 2014.

[3] 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.

[4] Romain Cosset. *Applications des fonctions thêta à la cryptographie sur courbes hyperelliptiques.* Theses, Université Henri Poincaré - Nancy I, November 2011.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik*, 1997(485):93–122, 1997.

[10] 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.

[11] 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.

[12] 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*, page 472–503, Berlin, Heidelberg, 2023. Springer-Verlag.

[13] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, oct 1997.

[14] Joseph H. Silverman. *The Geometry of Elliptic Curves*. Springer New York, New York, NY, 2009.

[15] Benjamin Andrew Smith. *Explicit endomorphisms and correspondences*. PhD thesis, University of Sydney, 2005-12-23.

[16] J. Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences, Série I*, 273:238–241, juillet 1971.

[17] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography, Second Edition*. Chapman & Hall/CRC, 2 edition, 2008.

Appendix A

Magma Code

The code acompanying this thesis can be found at the following Gitlab repository:

https://git.uwaterloo.ca/jmlaflam/sidh-attack/.

It contains the following files:

- `Gluing.m`: This file contains implementations of the methods in section 4.4.1 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 4.4.2 to convert theta points to points on a product of elliptic curves.
- `automorphism.m`: This file contains implementations of the methods in section 4.4.3 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`: This file is a modified version of the file `Image.m` of the `Avisogenies` package which contains the modifications described in section 4.3.
- `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 5.1 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 can be downloaded 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.